Jérôme Darmont
Boris Novikov
Robert Wrembel (Eds.)

Communications in Computer and Information Science 1259

New Trends in Databases and Information Systems

ADBIS 2020 Short Papers
Lyon, France, August 25–27, 2020
Proceedings

Springer

Jérôme Darmont · Boris Novikov ·
Robert Wrembel (Eds.)

New Trends in Databases and Information Systems

ADBIS 2020 Short Papers
Lyon, France, August 25–27, 2020
Proceedings

Springer

Editors
Jérôme Darmont
Université Lumière Lyon 2
Lyon, France

Robert Wrembel
Poznań University of Technology
Poznań, Poland

Boris Novikov
National Research University
Higher School of Economics
St. Petersburg, Russia

ISSN 1865-0929 ISSN 1865-0937 (electronic)
Communications in Computer and Information Science
ISBN 978-3-030-54622-9 ISBN 978-3-030-54623-6 (eBook)
https://doi.org/10.1007/978-3-030-54623-6

Preface

The 24th European Conference on Advances in Databases and Information Systems (ADBIS 2020) was set to be held in Lyon, France, during August 25–28, 2020, in conjunction with the 24th International Conference on Theory and Practice of Digital Libraries (TPDL 2020) and the 16th EDA days on Business Intelligence & Big Data (EDA 2020). However, because of the worldwide COVID-19 crisis, ADBIS, TPDL, and EDA had to take place online during August 25–27, 2020. Yet, the three conferences joined their forces to propose common keynotes, workshops, and a Doctoral Consortium.

ADBIS established itself as a highly recognized conference in Europe. It aims at providing an international forum for exchanging research achievements on databases and data engineering, both from practical and theoretical perspectives. ADBIS also aims at promoting the interaction and collaboration of the database and data engineering as well as information systems research communities worldwide.

Previous ADBIS conferences were held in Saint Petersburg (1997), Poznan (1998), Maribor (1999), Prague (2000), Vilnius (2001), Bratislava (2002), Dresden (2003), Budapest (2004), Tallinn (2005), Thessaloniki (2006), Varna (2007), Pori (2008), Riga (2009), Novi Sad (2010), Vienna (2011), Poznan (2012), Genoa (2013), Ohrid (2014), Poitiers (2015), Prague (2016), Nicosia (2017), Budapest (2018), and Bled (2019).

ADBIS 2020 attracted 82 paper submissions, which were reviewed by an International Program Committee constituted of members from 28 countries. The Program Committee selected 13 long and 18 short research papers, which yields acceptance rates of 16% and 26%, respectively. The selected short papers span a wide spectrum of topics related to the ADBIS conference from different areas of research in database and information systems, including database performance, security, business intelligence, semantic technologies, social media analysis, and classification.

Finally, we would like to express our sincere gratitude to everyone who contributed to make ADBIS 2020 successful:

- authors, for submitting their research papers to the conference;
- keynote speakers who honored us with their talks at ADBIS 2020;
- members of the Program Committee and external reviewers, for dedicating their time and expertise to build a high-quality program;
- members of the ADBIS Steering Committee for their trust and support, and especially its chair Yannis Manolopoulos;
- our academic and private sponsors and partners: the Universities of Lyon 2 and Lyon 3, IDEXLYON, the ERIC laboratory, OnlyLyon, Springer for publishing these proceedings, the Coalition for Networked Information, as well as our PCO Insight Outside;

- last but not least, all members of the Organizing Committee, who had to switch from a physical organization to an online organization in troubled times.

June 2020

Jérôme Darmont
Boris Novikov
Robert Wrembel

Organization

General Chair

Jérôme Darmont	Université Lyon 2, France

Steering Committee Chair

Yannis Manolopoulos	Open University of Cyprus, Cyprus

Steering Committee

Ladjel Bellatreche	ENSMA Poitiers, France
Andras Benczur	Eötvös Loránd University, Hungary
Maria Bielikova	Slovak University of Technology in Bratislava, Slovakia
Barbara Catania	University of Genoa, Italy
Jérôme Darmont	Université Lyon 2, France
Johann Eder	Alpen Adria Universität Klagenfurt, Austria
Theo Haerder	Technical University Kaiserslautern, Germany
Mirjana Ivanović	University of Novi Sad, Serbia
Hannu Jaakkola	Tampere University of Technology, Finland
Marite Kirikova	Riga Technical University, Latvia
Rainer Manthey	University of Bonn, Germany
Manuk Manukyan	Yerevan State University, Armenia
Tadeusz Morzy	Poznan University of Technology, Poland
Boris Novikov	National Research University, Higher School of Economics in Saint Petersburg, Russia
George Papadopoulos	University of Cyprus, Cyprus
Jaroslav Pokorný	Charles University in Prague, Czech Republic
Boris Rachev	Technical University of Varna, Bulgaria
Sherif Sakr	University of Tartu, Estonia
Bernhard Thalheim	Christian Albrechts University, Germany
Goce Trajcevski	Iowa State University, USA
Valentino Vranić	Slovak University of Technology in Bratislava, Slovakia
Tatjana Welzer	University of Maribor, Slovenia
Robert Wrembel	Poznan Unviersity of Technology, Poland
Ester Zumpano	University of Calabria, Italy

Program Committee Chairs

Boris Novikov	National Research University, Higher School of Economics in Saint Petersburg, Russia
Robert Wrembel	Poznan University of Technology, Poland

Program Committee

Alberto Abelló	Universitat Politècnica de Catalunya, Spain
Bernd Amann	Sorbonne Université, France
Andreas Behrend	University of Bonn, Germany
Ladjel Bellatreche	ENSMA Poitiers, France
András Benczúr	Eötvös Loránd University, Hungary
Fadila Bentayeb	Université Lyon 2, France
Maria Bielikova	Slovak University of Technology in Bratislava, Slovakia
Sandro Bimonte	IRSTEA Clermont-Ferrand, France
Miklos Biro	Software Competence Center Hagenberg, Austria
Pawel Boiński	Poznan University of Technology, Poland
Omar Boussaid	Université Lyon 2, France
Drazen Brdjanin	University of Banja Luka, Serbia
Albertas Caplinskas	Vilnius University, Lithuania
Barbara Catania	University of Genoa, Italy
Tania Cerquitelli	Politecnico di Torino, Italy
Ajantha Dahanayake	Lappeenranta University of Technology, Finland
Jérôme Darmont	Université Lyon 2, France
Christos Doulkeridis	University of Piraeus, Greece
Johann Eder	Alpen Adria Universität Klagenfurt, Austria
Markus Endres	University of Passau, Germany
Werner Esswein	Technische Universität Dresden, Germany
Georgios Evangelidis	University of Macedonia, Greece
Flavio Ferrarotti	Software Competence Centre Hagenberg, Austria
Flavius Frasincar	Erasmus University Rotterdam, The Netherlands
Johann Gamper	Free University of Bozen-Bolzano, Italy
Matteo Golfarelli	University of Bologna, Italy
Marcin Gorawski	Silesian University of Technology, Poland
Jānis Grabis	Riga Technical University, Latvia
Le Gruenwald	The University of Oklahoma, USA
Francesco Guerra	Università di Modena e Reggio Emilia, Italy
Giancarlo Guizzardi	Federal University of Espirito Santo, Brazil
Hele-Mai Haav	Tallinn University of Technology, Estonia
Tomas Horvath	Eötvös Loránd University, Hungary
Theo Härder	Technical University Kaiserslautern, Germany
Marko Hölbl	University of Maribor, Slovenia
Mirjana Ivanović	University of Novi Sad, Serbia
Hannu Jaakkola	University of Tampere, Finland

Stefan Jablonski	University of Bayreuth, Germany
Aida Kamišalić Latifić	Universtiy of Maribor, Slovenia
Dimitris Karagiannis	University of Vienna, Austria
Zoubida Kedad	University of Versailles, France
Attila Kiss	Eötvös Loránd University, Hungary
Michal Krátký	Technical University of Ostrava, Czech Republic
Julius Köpke	Alpen Adria Universität Klagenfurt, Austria
Dejan Lavbič	University of Ljubljana, Slovenia
Wolfgang Lehner	Technical University Dresden, Germany
Daniel Lemire	Université du Québec à Montréal, Canada
Sebastian Link	The University of Auckland, New Zealand
Yurii Litvinov	Saint Petersburg State University, Russia
Federica Mandreoli	University of Modena, Italy
Yannis Manolopoulos	Open University of Cyprus, Cyprus
Manuk Manukyan	Yerevan State University, Armenia
Patrick Marcel	Université de Tours, France
Bálint Molnár	Eötvös University of Budapest, Hungary
Angelo Montanari	University of Udine, Italy
Tadeusz Morzy	Poznan University of Technology, Poland
Martin Nečaský	Charles University, Czech Republic
Boris Novikov	National Research University, Higher School of Economics in Saint Petersburg, Russia
Kjetil Nørvåg	Norwegian University of Science and Technology, Norway
Andreas Oberweis	Karlsruhe Institute of Technology, Germany
Carlos Ordonez	University of Houston, USA
George Papadopoulos	University of Cyprus, Cyprus
András Pataricza	Budapest University of Technology and Economics, Hungary
Jaroslav Pokorný	Charles University in Prague, Czech Republic
Giuseppe Polese	University of Salerno, Italy
Alvaro E. Prieto	University of Extremadura, Spain
Miloš Radovanović	University of Novi Sad, Serbia
Heri Ramampiaro	Norwegian University of Science and Technology, Norway
Franck Ravat	Université de Toulouse, France
Stefano Rizzi	University of Bologna, Italy
Oscar Romero	Universitat Politècnica de Catalunya, Spain
Gunter Saake	University of Magdeburg, Germany
Kai-Uwe Sattler	Technical University Ilmenau, Germany
Milos Savic	University of Novi Sad, Serbia
Patrick Schäfer	Humboldt Universität zu Berlin, Germany
Timos Sellis	Swinburne University of Technology, Australia
Bela Stantic	Griffith University, Australia
Kostas Stefanidis	University of Tampere, Finland
Sergey Stupnikov	Russian Academy of Sciences, Russia

Olivier Teste	Université de Toulouse, France
Bernhard Thalheim	Christian Albrechts University Kiel, Germany
Goce Trajcevski	Iowa State University, USA
Raquel Trillo-Lado	Universidad de Zaragoza, Spain
Olegas Vasilecas	Vilnius Gediminas Technical University, Lithuania
Goran Velinov	UKIM, North Macedonia
Isabelle Wattiau	ESSEC, CNAM, France
Tatjana Welzer	University of Maribor, Slovenia
Marek Wojciechowski	Poznan University of Technology, Poland
Robert Wrembel	Poznan University of Technology, Poland
Anna Yarygina	Saint Petersburg State University, Russia
Vladimir Zadorozhny	University of Pittsburgh, USA
Jaroslav Zendulka	Brno University of Technology, Czech Republic

Additional Reviewers

Andrea Brunello
Alexandre Chanson
Stefano Cirillo
Vincenzo Deufemia
Senén González
Anna Gorawska
Sergio Ilarri
Igor Kuralenok
Petar Jovanovic
Nicolas Labroche
Christos Mettouris
Sergi Nadal
Demetris Paschalides
Oszkár Semeráth
Jakub Ševcech
Artem Trofimov
Willeme Verdeaux
Alexandros Yeratziotis

Proceeding Chairs

Fadila Bentayeb	Université Lyon 2, France
Elöd Egyed-Zsigmond	INSA Lyon, France
Nadia Kabachi	Université Lyon 1, France

Workshop Chairs

Ladjel Bellatreche	ENSMA Poitiers, France
Mária Bieliková	Slovak University of Technology, Slovakia

Christos Papatheodorou	Ionian University, Greece
Guilaine Talens	Université Lyon 3, France

Doctoral Consortium Chairs

Barbara Catania	University of Genoa, Italy
Elena Demidova	L3S Research Center, Germany
Oscar Romero	Universitat Politècnica de Catalunya, Spain
Maja Zumer	University of Ljubljana, Slovenia

Journal Special Issue Chair

Ladjel Bellatreche	ENSMA Poitiers, France

Publicity Chair

Selma Khouri	ESI Alger, Algeria

Organizing Committee

Fadila Bentayeb	Université Lyon 2, France
Omar Boussaïd	Université Lyon 2, France
Jérôme Darmont	Université Lyon 2, France
Fabien Duchateau	Université Lyon 1, France
Elöd Egyed-Zsigmond	INSA Lyon, France
Mihaela Juganaru-Mathieu	École des Mines de Saint-Étienne, France
Nadia Kabachi	Université Lyon 1, France
Omar Larouk	ENSSIB Lyon, France
Fabrice Muhlenbach	Université de Saint-Étienne, France
Habiba Osman	Université Lyon 2, France
Muriel Perez	Université de Saint-Étienne, France
Pegdwendé Sawadogo	Université Lyon 2, France
Guilaine Talens	Université Lyon 3, France
Caroline Wintergerst	Université Lyon 3, France

Contents

Data Access and Database Performance

ARTful Skyline Computation for In-Memory Database Systems

Maximilian E. Schüle(✉), Alex Kulikov, Alfons Kemper, and Thomas Neumann

Technical University of Munich, Munich, Germany
{m.schuele,alex.kulikov,alfons.kemper,thomas.neumann}@tum.de

Abstract. Skyline operators compute the Pareto-optimum on multi-dimensional data inside disk-based database systems. With the arising trend of main-memory database systems, pipelines process tuples in parallel and in-memory index structures, such as the adaptive radix tree, reduce the space consumption and accelerate query execution.

We argue that modern database systems are well suited to progressive skyline operators. In addition, space-efficient index structures together with tree-based skyline algorithms improve the overall performance on categorical input data. In this work, we parallelise skyline algorithms, reduce their memory consumption and allow their integration into the main-memory database system HyPer. In our evaluation, we show that our parallelisation techniques scale linearly with every additional worker, and that the adaptive radix tree reduces memory consumption in comparison to existing tree-based approaches for skyline computation.

Keywords: Skyline operator · In-Memory DBMS · Adaptive radix tree

1 Introduction

The skyline algorithm finds interesting tuples within multi-dimensional data sets. Specifically, these tuples form the Pareto-optimal set that contains only the best tuples regarding all criteria. Formally, the output set is defined as all tuples that are not dominated by any other tuple of the input set.

From a theoretical point of view, computing the skyline of a set of tuples corresponds to the mathematical problem of finding the maxima of a set of vectors [8]. A vector $p \in \mathbb{R}^n$ dominates another vector $q \in \mathbb{R}^n$ if p is at least as good as q in every dimension, and superior in at least one:

$$p \succ q \Leftrightarrow \forall i \in [n].p[i] \succeq q[i] \wedge \exists j \in [n].p[j] \succ q[j]. \quad (1)$$

Börzsönyi et al. [2] provided the first skyline implementations and an SQL extension (see Listing 1.1). Their work compared the algorithm to a skyline formed out of skyscrapers: only those are visible which are either closer (regarding the distance) or higher than any other to a specific viewing point. Even though SQL-92 is capable of expressing skyline queries (see Listing 1.2), the authors proposed an integration as an operator that executes optimised skyline algorithms within the database system.

```
SELECT * FROM inputtable i WHERE ... GROUP BY ... HAVING ...
SKYLINE OF [DISTINCT] d1 [MIN | MAX], ... , dn [MIN | MAX]
ORDER BY ...
```

Listing 1.1. Skyline extension of SQL: d_1, ... , d_n are the dimensions; MIN and MAX specify whether each dimension has to be minimised or maximised.

```
SELECT * FROM inputtable q WHERE NOT EXISTS (
  SELECT * FROM inputtable p WHERE p.d1 <= q.d1 AND ... AND p.dn<=q.dn
     AND (p.d1 < q.d1 OR ... OR p.dn<q.dn ))
```

Listing 1.2. Skyline query in SQL on a table *inputtable* with attributes d_1, ... , d_n.

We argue that integrated skyline operators can benefit from modern database systems that offer in-memory index structures as well as pipelined tuple processing. In the following, we integrate the naive-nested-loops skyline algorithm as an operator into the main-memory database system HyPer [6]. As a native operator, it supports code-generation according to the producer-consumer concept, that pushes tuples towards the parent operator in parallel pipelines. This enables the parallelisation of progressive skyline algorithms that continuously produce output. Furthermore, we optimise the space requirements for trie-based skyline algorithms and parallelise all introduced implementations for multi-threaded execution. In summary, this work's contributions are:

- the integration of a skyline operator into the main-memory database system HyPer following the producer-consumer model,
- a memory reduction for trie-based skyline computation on categorical data due to the usage of the adaptive radix tree,
- the parallelisation of naive-nested-loops as well as tree-based skyline algorithms within the context of database systems,
- and an evaluation in terms of run time, memory usage and scalability that compares naive-nested-loops to tree-based skyline algorithms.

This work is organised as follows: First, we give an overview of the underlying main-memory database system with its adaptive radix tree and existing skyline algorithms. Hereafter, this paper proposes a novel skyline algorithm called SARTS, which uses the adaptive radix tree for dominance checks. Afterwards, parallelisation techniques are first explained and then applied to the given skyline algorithms. For the evaluation, we vary the number of input tuples as well as the number of available threads.

2 Related Work

As this work combines main-memory database systems with skyline algorithms, this section introduces the underlying operator concepts within modern database systems and common skyline algorithms.

2.1 Main-Memory Database Systems

HyPer [6,13,14] is an in-memory database system that introduced code-generation according to the producer-consumer model. Instead of traditional Volcano-style query execution [4], where the topmost operator iteratively demands the underlying ones to return tuples, operators in HyPer push tuples towards the parent operator. Two functions, `produce()` and `consume()`, generate the corresponding code using the LLVM compiler framework. During code-generation, `produce()` is called recursively from top to bottom, then each call evokes a `consume()` call on the parent node. This generates the code for processing tuples in parallel pipelines. We later integrate skyline as an operator that initiates parallel pipelines.

The adaptive radix tree (ART) [9] is the in-memory index-structure used in HyPer to retrieve tuples by their identifier. In contrast to a radix tree, the node's size is adaptive in order to reduce the memory consumption and improve the caching performance. The ART offers four different node types for either four, 16, 48 or 256 keys and can replace various tries such as prefix-trees represented by radix trees.

2.2 Skyline Algorithms

The naive-nested-loops (NNL) algorithm from the original paper [2] forms the basis for our in-database implementation A nested loop compares each tuple to each other one whether it is not dominated and therefore forms part of the return set. To reduce the number of disk accesses, the block-nested-loops (BNL) algorithm maintains a window in main-memory of all tuples considered for the skyline to that point. The divide-and-conquer (DNC) algorithm partitions [16] the tuples recursively and performs dominance checks when merging partitions.

Since its invention, skyline algorithms have been based on different data structures and hardware [5]. This work mainly incorporates research on the parallelisation of skyline algorithms [7,10,15,17] that produce progressive output [11]. This facilitates the integration into database systems according to the producer-consumer model.

Sorting-based algorithms such as Sort-Filter-Skyline (SFS) [3] or SaLSa [1] pre-sort the input first before computing the skyline. Pre-sorted input allows elements to be pruned which are worse. The skyline-using-tree-sorting (ST-S) algorithm [12] is tuned for binary attribute values, as it stores tuples in a radix tree called *N-tree* to perform dominance checks. In this work, we extend the algorithm to support categorical data and replace the N-tree with the ART.

3 SARTS

This section presents SARTS (Skyline using ART Sorting-based), a novel skyline algorithm for categorical attributes. It improves the core concepts of ST-S by implementing a more efficient indexing structure for dominance checks—the

ART. As our proposed SARTS algorithm is based on the ST-S algorithm, we first explain the extension of ST-S for categorical attributes, before we proceed with the integration of the adaptive radix tree.

3.1 ST-S for Categorical Attributes

Every inner node of the N-tree in the ST-S algorithm, including the root, has an array, which can hold as many children as there are possible attribute values. Each path taken from the root to a leaf represents a tuple, which is assigned a score. The score of a particular tuple t with n attributes is determined by a scoring function with $t[i]$ as the i-th attribute of the tuple:

$$score(t) := \sum_{i=0}^{i<n} 2^{n-i} \cdot t[i]. \tag{2}$$

In each inner node, `minScore` and `maxScore` mark the boundaries for the tuple's possible score within branches descending from this node.
At the beginning, a monotonic function `minC()` or `maxC()` defines an order:

$$minC(t) := \big(\min_{0 \leq i < n} (t[i]), \sum_{i=0}^{i<n} t[i] \big). \tag{3}$$

It consists of two components: a main comparison attribute, which is the smallest value of all tuple's attributes, and a tie-breaker that is the sum of all the tuple's attribute values. The ST-S algorithm (Algorithm 1) works as follows:

1. The tuples are presorted with `minC()` (line 1).
2. The threshold tuple t_{stop}, undefined at the beginning, is later updated (lines 11–12) with knowledge of tuples that are part of the skyline.
3. The first tuple t_0 from the presorted data set is always part of the skyline. It gets inserted into the tree and is put out as part of the skyline (lines 3–5).
4. The following loop checks for every input tuple t whether it is dominated by any tuple already in the skyline (line 8). The checks are carried out with the help of the tree, which holds all the skyline tuples to date.
5. If a tuple t is dominated by some other tuple in the skyline, it is no longer considered (line 8). Otherwise, it is inserted into the tree (line 9), so that it is able to eliminate future, dominated tuples.
6. If the maximum attribute value of the new skyline tuple t is smaller than the maximum attribute value of the threshold t_{stop}, then the threshold is updated (line 12), and now holds the value of t, until the next update occurs.
7. The algorithm stops as soon as all tuples left in the data set are a priori dominated by the threshold (line 7).
8. If the maximum attribute value of the threshold tuple t_{stop} is less than or equal to the minimum attribute value of the current tuple t, then none of the remaining, sorted tuples are part of the skyline.

Both the `insert()` and `is_dominated()` operations have been slightly modified from the original paper to deal with categorical attributes rather than binary ones.

Algorithm 1. ST-S Algorithm

```
    Input: Tuple List $T$, Tree $tree$
    Output: Skyline $skyline$
 1: Sort $T$ in-place using a monotonic function minC()
 2: $t_0 \leftarrow$ first element of $T$
 3: $t_{stop} \leftarrow t_0$
 4: insert($t_0$, $tree.root$, 0)
 5: Add $t_0$ to $skyline$ // $t_0$ always part of skyline due to presorting
 6: for each tuple $t \in T \setminus \{t_0\}$ do
 7:     if $max(t_{stop}) \leq min(t)$ and $t_{stop} \neq t$ then return
 8:     if not is_dominated($t$, $tree.root$, 0, $score(t)$) then
 9:         insert($t$, $tree.root$, 0)
10:         Add $t$ to $skyline$
11:         if $max(t) < max(t_{stop})$ then
12:             $t_{stop} \leftarrow t$
```

3.2 ART for Skyline

The interface of the ART has been kept similar to that of the N-Tree in ST-S. This enables the very straightforward integration of the ART into the algorithm, because the `insert()` and `is_dominated()` operations still have the same signature as in ST-S. While `insert()` is slightly different from the original variant, the `is_dominated()` operation is almost identical to the one in ST-S. The `insert()` operation for SARTS differs from the ST-S variant in this both finding the correct child to the current node and creating a new child are outsourced into two separate functions: `findChild()` and `newChild()`. In addition to that, before a new child can be created, the current node might first need to `grow()` to the next-bigger type, in order to create space for the new child. The pseudo-code to the `insert()` operation is given in Algorithm 2.

The main difference within the `is_dominated()` operation is, similarly to `insert()`, that it uses `findChild()` to determine the correct child for further traversal.

In addition to that, just like the nodes of the N-Tree, the inner nodes of the ART have to be extended by a `minScore` and a `maxScore`, and the leaf nodes by the `score` attribute and an array of `tupleIDs`. This enables the faster traversing of the tree during dominance checks, by skipping tree regions that cannot dominate the current tuple.

4 Parallelisation

The following section presents the parallelisation approaches for traditional skyline algorithms, such as NNL and DNC[1], as well as for two of the newer algorithms: ST-S and SARTS[2]. The corresponding source-code is publicly available.

[1] https://gitlab.db.in.tum.de/alex_kulikov/skyline-computation.
[2] https://gitlab.db.in.tum.de/alex_kulikov/skyline-categorical.

Algorithm 2. INSERT Operation for SARTS

```
    Input: Tuple $t$, Node $parent$ Node $current$, Level $level$, Attributes $atts$
 1: if $level = 0$ then
 2:     $node.minScore \leftarrow 0$
 3:     $node.maxScore \leftarrow \sum_{i=0}^{t.size-1}(2^{t.size-i} \cdot max(atts))$
 4: else if $level$ != $t.size$ then
 5:     $node.minScore \leftarrow \sum_{i=0}^{level-1}(2^{t.size-i} \cdot t[i])$
 6:     $node.maxScore \leftarrow node.minScore + \sum_{i=level}^{t.size-1}(2^{t.size-i} \cdot max(atts))$
 7: if $level = t.size$ then
 8:     $node.score \leftarrow score(t)$
 9:     Append $t.tupleID$ to $node.tupleIDs$
10: else
11:     $child \leftarrow findChild(current, t[level])$
12:     if $child$ is $None$ then
13:         if $current.size$ != 256 then
14:             $grow(parent, current, t[level-1])$
15:         $child \leftarrow newChild(current, t[level])$
16:     insert($t$, $current$, $child$, $level + 1$)
```

4.1 Naive-/Block-Nested-Loops

The main idea when parallelising the naive-nested-loops algorithm is to use the `parallel_for` construct for the outer loop of the algorithm. The inner loop could also be taken for this purpose, but then the code, which finds itself in the outer loop but not in the inner one would be running sequentially, thus reducing the benefit of parallelising the code in the first place. The pseudo-code notation of the parallelised version of naive-nested-loops is given in Algorithm 3.

Algorithm 3. Parallel NNL

```
    Input: Tuple List $T$
    Output: Skyline $skyline$
 1: parallel_for each tuple $t \in T$ do
 2:     $is\_not\_dominated \leftarrow$ True
 3:     for each tuple $d \in T \setminus \{t\}$ do
 4:         if dominates($d$, $t$) then
 5:             $is\_not\_dominated \leftarrow$ False
 6:             break
 7:     if $is\_not\_dominated$ then
 8:         Add $t$ to $skyline$
```

4.2 Divide-and-Conquer

Two different parallelisation techniques were applied to the divide-and-conquer algorithm. Instead of applying a sequential sorting algorithm, `parallel_sort` sorts the elements using several worker threads simultaneously, and thus produces the result significantly faster than sequential functions for large data sets. `parallel_sort` is applied in two places within the DNC algorithm:

1. *Finding the median.* After sorting the tuples, the median of the data set is taken to be the element located exactly in the middle of the sorted set.

2. *Determining the minimum for two dimensions.* The skyline can be computed by finding the minimum of the first subset and comparing it to all elements of the second subset.

4.3 SARTS and ST-S

The parallelising of the ST-S and SARTS algorithms results in almost identical implementations. As the interfaces of both trees are technically the same, the algorithms were also parallelised via the same approach.

The main idea is to divide the original data set into as many partitions as there are threads on the machine. One thread for each partition computes the skyline of its tuples. Every thread receives its own tree structure to store the tuples that are part of the skyline and to perform dominance checks. In other words, the sequential version of SARTS (resp. ST-S) is simultaneously applied to each of the partitions. As soon as the skyline of every partition has been computed, the resulting skylines are merged to produce the final one. The skylines of all partitions combined are much smaller than the original data set. Therefore, the final merge does not take as much time as computing the entire skyline from scratch.

In addition to the main parallelisation approach, presorting the tuples also happens in parallel before the actual algorithm begins. As the original data set tends to be very large in real-world applications, sorting it in parallel leads to a very significant efficiency boost.

The skyline is computed similarly to the non-parallelised version, with one major difference. Whenever a tuple that is definitely part of the skyline is stored, it is not merely appended to some list of skyline tuples. Instead, it is stored into a common `sub_results` array, to which all skyline threads share access.

5 Evaluation

This section discusses the evaluation of the following algorithms: naive-nested-loops (NNL), block-nested-loops (BNL), divide-and-conquer (DNC), ST-S and SARTS. All the tests were conducted on a Linux Mint 18.2 machine offering an Intel Core i7-5500U CPU with a 4096 KB cache and 8 GB DDR3L of main-memory. As the tree-based skyline algorithms, such as ST-S and SARTS, are restricted to categorical attributes, tests that include the algorithms ST-S and SARTS were conducted using a limited set of integers as categories, ranging from 0 to 255. All other tests were performed with continuous attributes, represented as `double` values.

5.1 Non-progressive Algorithms

The non-progressive skyline types included in this work are the block-nested-loops and the divide-and-conquer algorithms. In all three of the conducted tests, BNL scales significantly better than DNC. It shows overall better performance with an increasing number of tuples, dimensions and also threads (Fig. 1). Herewith, the results are similar to the ones produced in the original paper [2], which introduced BNL and DNC.

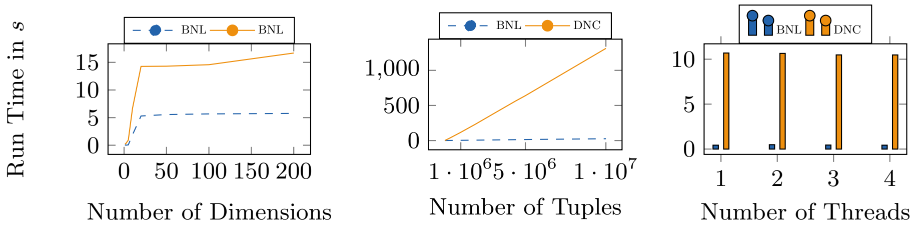


Fig. 1. Run time of non-progressive algorithms by number of tuples (default: 5 dimensions, 256 categories, 4 threads and 10,000 input tuples).

5.2 Progressive Algorithms

Naive-nested-loops can be both progressive and parallelisable and therefore compared to the two newer algorithms ST-S and SARTS. As expected, for a rising number of tuples, both ST-S and SARTS perform extremely well. As shown in Fig. 2, they significantly outperform naive-nested-loops with larger input. This is not surprising, as ST-S and SARTS were specifically developed for large categorical data sets. It is due to the efficient nature of the tree structures used that dominance checks can be conducted very efficiently, and depend less on the number of tuples than on the dimensionality of the data set.

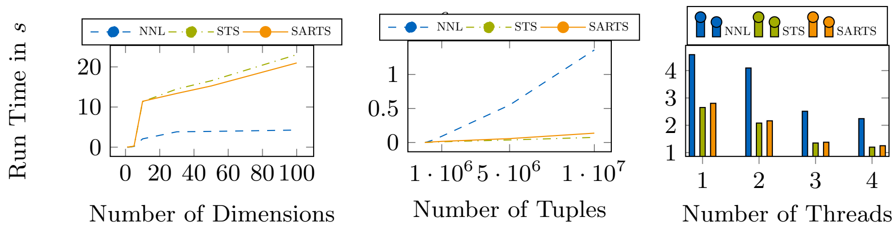


Fig. 2. Run time of progressive algorithms by number of tuples (default: 5 dimensions, 256 categories, 4 threads and 10,000 input tuples).

When looking at the results of scaling with dimensionality, the naive-nested-loops algorithm significantly outperforms both parallelised and sequential versions of ST-S and SARTS. The reason for this is that radix-based tree structures—N-Tree and ART—generally scale badly with longer keys. This is the trade-off they have to accept for very efficient scaling with the number of inserted elements. The longer the keys of the data set are, the higher the tree gets, and the longer it takes to traverse the tree from top to bottom. In the application area of skyline computation, the length of a key corresponds to the dimensionality of a tuple. Hence, the more dimensions the tuples of a data set have, the less efficient tree-based dominance checks become.

A comparison of progressive algorithms depending on the number of threads available shows that both ST-S as well as SARTS outperform naive-nested-loops. As expected, all parallelised algorithms scale with the number of available threads.

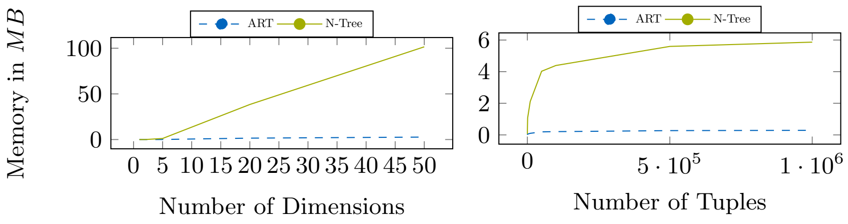


Fig. 3. Memory usage of ART and N-Tree by dimensionality and tuples (256 categories, 4 threads; left: 1000 input tuples, right: 5 dimensions).

Memory Usage. The last two tests compare the main-memory usage of the ART to that of the N-Tree. Figure 3 shows that the ART significantly consumes less space than the N-Tree. The memory consumption is similar when using multiple dimensions: While the ART already performs better than the N-Tree for low number of dimensions, it generally scales much more efficiently with high dimensionality. Thus, it can be concluded that the SARTS algorithm is significantly more memory-efficient than ST-S due to the usage of the ART.

6 Conclusion

This work has integrated skyline algorithms as an operator inside the main-memory database system HyPer according to the producer-consumer model. As in-memory index structures improve look-up performance in main-memory database systems, we replaced traditional radix trees by the adaptive radix tree for fast skyline computation on categorical data. This called SARTS algorithm displayed the same lookup performance as its ancestor algorithm, ST-S, but was superior with regard to space consumption, due to adaptive nodes. We successfully parallelised naive-nested-loops, divide-and-conquer and the tree-based algorithms to allow scaling to multiple cores.

References

1. Bartolini, I., Ciaccia, P., Patella, M.: Efficient sort-based skyline evaluation. ACM Trans. Database Syst. **33**(4) (2008). https://doi.org/10.1145/1412331.1412343

2. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, Heidelberg, Germany, 2–6 April 2001. IEEE Computer Society (2001). https://doi.org/10.1109/ICDE.2001.914855
3. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: ICDE, Bangalore, India, 5–8 March 2003. IEEE Computer Society (2003). https://doi.org/10.1109/ICDE.2003.1260846
4. Graefe, G.: Encapsulation of parallelism in the volcano query processing system. In: SIGMOD, Atlantic City, NJ, USA, 23–25 May 1990. ACM Press (1990). https://doi.org/10.1145/93597.98720
5. Hose, K., Vlachou, A.: A survey of skyline processing in highly distributed environments. VLDB J. **21**(3) (2012). https://doi.org/10.1007/s00778-011-0246-6
6. Hubig, N., Passing, L., Schüle, M.E., Vorona, D., Kemper, A., Neumann, T.: HyPerInsight: data exploration deep inside hyper. In: CIKM, Singapore, 06–10 November 2017 (2017). https://doi.org/10.1145/3132847.3133167
7. Köhler, H., Yang, J., Zhou, X.: Efficient parallel skyline processing using hyperplane projections. In: SIGMOD, Athens, Greece, 12–16 June 2011. ACM (2011). https://doi.org/10.1145/1989323.1989333
8. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. J. ACM **22**(4) (1975). https://doi.org/10.1145/321906.321910
9. Leis, V., Kemper, A., Neumann, T.: The adaptive radix tree: artful indexing for main-memory databases. In: ICDE, Brisbane, Australia, 8–12 April 2013. IEEE Computer Society (2013). https://doi.org/10.1109/ICDE.2013.6544812
10. Liknes, S., Vlachou, A., Doulkeridis, C., Nørvåg, K.: APSkyline: improved skyline computation for multicore architectures. In: DASFAA, Bali, Indonesia, 21–24 April 2014 (2014). https://doi.org/10.1007/978-3-319-05810-8_21
11. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: SIGMOD, San Diego, California, USA, 9–12 June 2003. ACM (2003). https://doi.org/10.1145/872757.872814
12. Rahman, M.F., Asudeh, A., Koudas, N., Das, G.: Efficient computation of subspace skyline over categorical domains. In: CIKM, Singapore, 06–10 November 2017. ACM (2017). https://doi.org/10.1145/3132847.3133012
13. Schüle, M., Bungeroth, M., Vorona, D., Kemper, A., Günnemann, S., Neumann, T.: ML2SQL - compiling a declarative machine learning language to SQL and python. In: EDBT, Lisbon, Portugal, 26–29 March 2019 (2019). https://doi.org/10.5441/002/edbt.2019.56
14. Schüle, M., et al.: The power of SQL lambda functions. In: EDBT, Lisbon, Portugal, 26–29 March 2019 (2019). https://doi.org/10.5441/002/edbt.2019.49
15. Tang, M., Yu, Y., Aref, W.G., Malluhi, Q.M., Ouzzani, M.: Efficient parallel skyline query processing for high-dimensional data. In: ICDE, Macao, China, 8–11 April 2019. IEEE (2019). https://doi.org/10.1109/ICDE.2019.00251
16. Vlachou, A., Doulkeridis, C., Kotidis, Y.: Angle-based space partitioning for efficient parallel skyline computation. In: SIGMOD, Vancouver, BC, Canada, 10–12 June 2008. ACM (2008). https://doi.org/10.1145/1376616.1376642
17. Zois, V., Gupta, D., Tsotras, V.J., Najjar, W.A., Roy, J.: Massively parallel skyline computation for processing-in-memory architectures. In: PACT, Limassol, Cyprus, 01–04 November 2018. ACM (2018). https://doi.org/10.1145/3243176.3243187

Quantum Computation and Its Effects in Database Systems

Szabolcs Jóczik and Attila Kiss(✉)

Faculty of Informatics, Eötvös Loránd University, Budapest 1117, Hungary
{joczikszabi,kiss}@inf.elte.hu
https://www.elte.hu/en/faculties/informatics

Abstract. Classically, searching an unsorted database requires a linear search, which is $\mathcal{O}(n)$ in time. Using Grover's quantum search algorithm, it is possible to do it in $\mathcal{O}(\sqrt{n})$ time which is a quadratic speedup compared to its classical counterpart. The aim of this research is to exploit this speedup and find other applications in different algorithms commonly used in database systems.

Keywords: Quantum computing · Grover's algorithm · Database operations

1 Introduction

Today's computers—both in theory (Turing machines) and practice (PCs, laptops, tablets, smartphones, ...)—are based on classical physics. They are limited by locality therefore. A quantum system can be in a superposition of many different states at the same time, and can exhibit interference effects during the course of its evolution. Moreover, spatially separated quantum systems may be entangled with each other and operations may have "non-local" effects because of this. Quantum computation is the field that investigates the computational power and other properties of computers based on quantum-mechanical principles. An important objective is to find quantum algorithms that are significantly faster than any classical algorithm solving the same problem [1]. Section 3 delivers a basic introduction to Grover's search algorithm and provides insight into how it can be used in the later chapters for database set operations. Section 4–7 proposes new algorithms that could be alternatively used in database systems. More specifically we propose a quantum algorithm for each of the following database set operations (Intersection, Set difference, Union, Projection). The majority of these algorithms are built upon the intersection operation using Grover's search algorithm. The validation is evaluated on both classical and quantum computers using IBM-Q in Sect. 8. Lastly in Sect. 9 we summarise the results we gathered using the proposed algorithms and make notes on some possible future improvements.

The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 13–23, 2020.
https://doi.org/10.1007/978-3-030-54623-6_2

2 Related Works

Applications of existing quantum algorithms in different fields of researches have been a center of attention for many years now. Researchers have been working on developing new algorithms and finding interesting applications of them that could replace their classical counterparts in the future. There are many examples like Shor's algorithm [2] in the field of prime factorization or Grover's quantum search algorithm [3] that has applications in cryptography [5], collision problems [6] and many more [7,8].

The motivation in this research paper was to find new applications of Grover's algorithm in database systems for basic set operations. Set operations such as intersection, difference, union, projection are fundamental building blocks in database queries that require fast and efficient algorithms. In this paper we are proposing four quantum algorithms for each of the previously mentioned operations that could possibly help in working with queries consisting of multiple operations. First we develop an algorithm for the intersection operation, then we use that algorithm to design the algorithms for set difference, union and projection. A similar work has been done before by Pang, C. Y., Zhou, R. G., Ding, C. B., and Hu, B. Q. [4]. They presented a quantum algorithm for the intersection operation with a running time of $\mathcal{O}(\sqrt{|A| \times |B| \times |C|}$, where $C = A \cap B$. The algorithm they developed is a combination of Grover's algorithm, classical memory and classical iterative computation that could be used as a starting point to develop algorithms for the other set operations as well but they were not presented in the paper. In another related work by Salman, T., and Baram, Y. [9] a similar approach was used in developing a quantum algorithm for the intersection operation.

3 Unstructured Database Search with Grover's Algorithm

Using Grover's search algorithm it is possible to find a specific item within a randomly ordered database of n items using $\mathcal{O}(\sqrt{n})$ operations (with probability $>\frac{1}{2}$). By contrast, a classical computer would require $\mathcal{O}(n)$ operations to achieve this, therefore, Grover's algorithm provides a quadratic speedup over an optimal classical algorithm and a good starting point to find applications in unstructured database systems.

3.1 Problem Definition

Suppose there's an unstructured database of n items that we map to the set $\{0, 1, \ldots, n-1\}$. Among these items there is one item with a unique property ω that we wish to locate. A common way to encode such a list is in terms of a function $f : \{0,1\}^N \longrightarrow \{0,1\}$, where $n = 2^N$ defined as

$$f(x) = \begin{cases} 1 & \text{if } x = \omega \\ 0 & \text{if } x \neq \omega, \end{cases}$$

To use a quantum computer for this problem, we encode the function into a unitary matrix called an oracle. First we choose a binary encoding of the items $x, \omega \in \{0,1\}^N$, thus we can represent each item using N qubits on a quantum computer. We then define the oracle matrix U_ω to act on any of the simple, standard basis states $|x\rangle$ by $U_\omega |x\rangle = (-1)^{f(x)} |x\rangle$.

We see that if x is an unmarked item, the oracle leaves the state unaffected. However, when we apply the oracle to the basis state $|\omega\rangle$, it maps $U_\omega |\omega\rangle = - |\omega\rangle$.

3.2 Algorithm

The initialization is carried out by applying a Hadamard transform on the system to achieve a uniform superposition of all states as follows

$$|0\rangle^{\otimes N} |1\rangle \longrightarrow \left(\frac{1}{\sqrt{2^N}} \sum_{x=0}^{2^N-1} |x\rangle \right) \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

The second step is the Grover Iteration, which is repeated $\lfloor \frac{\pi}{4}\sqrt{n} \rfloor$ times to achieve optimal probability. The iteration consists of the following two steps:

1.) Apply the oracle operator U_ω.
2.) Apply the Grover diffusion operator

$$G = 2|\psi\rangle\langle\psi| - I$$

In the last step a classical measurement is performed to determine the result. Since all proposed algorithms are inherited from Grover's algorithm, the optimal number of iterations in all cases equal to $\lfloor \frac{\pi}{4}\sqrt{n} \rfloor$.

We'll be also using the following function implicitly throughout the rest of the paper. Let $f : A \longrightarrow [0, \ldots, n-1]$ be an indexing function that maps the elements of A to the set $\{0, \ldots, n-1\}$. Now we can define a natural bijection between the subsets $X \subset A$ and the set of bitstrings with length n as follows

$$\begin{aligned} \mathcal{P}(A) &\longrightarrow \{0,1\}^n \\ X &\longmapsto [h(f^{-1}(0)), h(f^{-1}(1)), \ldots, h(f^{-1}(n-1))], \end{aligned}$$

where

$$h(x) = \begin{cases} 1 & x \in X \\ 0 & x \notin X. \end{cases}$$

4 Set Operation: Intersection

The intersection operator takes the results of two queries and returns only those records that appear in both result sets. A straightforward classical algorithm would be to go through each element of one of the sets and check whether it is in the second set or not. This algorithm would give us a $\mathcal{O}(nm)$ efficiency.

In this paper we assume that the given sets are unstructured and using sorting algorithms or hash tables is not allowed or computationally not worth it. With these assumptions and the help of Grover's algorithm, it is possible to find $A \cap B$ in $\mathcal{O}(\sqrt{2^N})$ computational time using the following proposed quantum.

4.1 Quantum Algorithm

Let A and B be two sets and let's assume that $n = |A| \leq |B|$. The intersection of A and B must be in the power set of A, thus $A \cap B \subset \mathcal{P}(A)$, where $|\mathcal{P}(A)| = 2^n$. Therefore we can use exactly n bits to represent all of the subsets of A, where in this case every element of A is represented by one qubit.

Now we can make use of Grover's search algorithm to search through the subsets $X \subset A$ and find $A \cap B$ with an appropriate Oracle function O_f, such as

$$O_f(X) = \begin{cases} 1 & X = A \cap B \\ 0 & X \neq A \cap B, \end{cases}$$

4.2 Implementation

As an example let $A = \{1, 2, 3\}$, $B = \{2, 3, 4\}$ and f defined as follows

$$f(1) = 0 \quad f(2) = 1 \quad f(3) = 2$$

We'll be using this definition of f for the rest of the paper. In this case the bitstring that represents $A \cap B$ is 011. The oracle function O_f needs to flag the value 011 and leave all other values unaltered. The implementation of the full algorithm including the oracle function can be seen on Fig. 1.

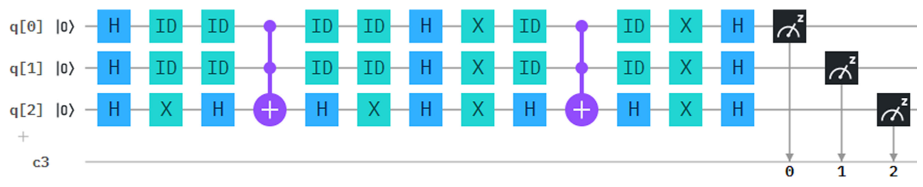


Fig. 1. Finding the intersection of A and B

5 Set Operation: Difference

The next set operation that we look at is the set difference which takes the results of the two queries and returns only those records of entries that appear only in the first query result. Although set difference operation by itself is required by only a few number of queries (i.e. `except`), it will be useful for the implementation of the set union operation in the next section. Similarly to the intersection operation, a straightforward classical algorithm would be to go through each element of the first query and check whether it is in the second set or not. If an element is not in the second query, then it is in $A \backslash B$, otherwise it's not. This algorithm would give us a $\mathcal{O}(nm)$ computational time, which is also the worst case.

Using the proposed quantum algorithm with the help of Grover's algorithm, we can reach an $\mathcal{O}(\sqrt{2^n})$ efficiency on a quantum computer, assuming that no sorting algorithms were used.

5.1 Quantum Algorithm

It is possible to obtain $A \backslash B$ using the intersection algorithm by simplifying the problem as $A \backslash B = A \backslash (A \cap B)$, where $A \backslash (A \cap B)$ can be viewed as the complement set of $(A \cap B)$ relative to A. Therefore the elements of $A \backslash B$ are going to be represented by those bits exactly that have zero values in the bitstring representation of $A \cap B$. Thus by flipping each of the bits' values, which can be achieved by applying X gates as the last step, we can get $A \backslash B$ from $A \cap B$ (Fig. 2).

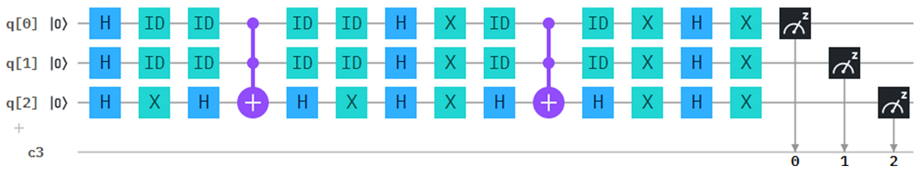


Fig. 2. Finding the difference of A and B

6 Set Operation: Union

6.1 Quantum Algorithm

First we need to find a partition of $A \cup B$, thus decomposing it into a union of distinct sets

$$A \cup B = (A \backslash B) \cup B,$$

then concatenating the two distinct sets afterwards. Since concatenation can be done efficiently on a classical computer, the speed up comes from calculating $A \backslash B$, which we already proposed an algorithm for. The elements of $A \backslash B$ and B are distinct and their union forms $A \cup B$, therefore it is convenient to use the sets $A \backslash B$ and B to construct $A \cup B$.

6.2 Implementation

The implementation is the same as for the set difference with an additional concatenation since $A \cup B$ is constructed using partition sets $(A \backslash B) \cup B$.

7 Set Operation: Projection

Database projection can be viewed as a function that takes a relation and a list of attributes of that relation as input and returns a relation containing only the specified list of attributes, thus the duplicate instances are removed. Therefore we need to deal with the following two independent problems.

7.1 Problem 1: Converting Multiset to Set

Since projection does not return duplicate attributes, we need to convert our resulting multiset into a set. Let's assume that the resulting multiset is

$$A = \{a_1^{m(a_1)}, a_2^{m(a_2)}, \ldots, a_n^{m(a_n)}\}.$$

We can further assume that the elements of A can be mapped to the set

$$X = \{0, 1, \ldots, n-1\} \text{ for some } n \in \mathbb{N}$$

Finding the distinct elements of A is equivalent to finding $X \cap A$.

7.2 Problem 2: Projection

The second problem is to acquire the elements of the specific column that we are looking for. First we use a binary coding of the table column name with length N, where $n = 2^N$. In this case n represents the number of columns. Let B_i be the i.th column and A_i the corresponding set of attributes that belong to that column, where $|A_i| = m = 2^M$ (Table 1).

Table 1. General database table

B_0	B_1	B_2	$\cdots$	B_n
a_{00}	a_{01}	a_{02}	$\cdots$	a_{0n}
a_{10}	a_{11}	a_{12}	$\cdots$	a_{1n}
a_{20}	a_{21}	a_{22}	$\cdots$	a_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_{m0}	a_{m1}	a_{m2}	$\cdots$	a_{mn}

The idea is to encode each element in the following state $|a_{ik}|b_i\rangle$ so we can use an oracle function to find exactly those particular elements that have the specific column encoded in its state. In order to do this we need to transform the elements into a superposition and use Grover's search algorithm for multiple entries as follows

$$\frac{1}{\sqrt{2^{M+N}}} \sum_i \sum_k |a_{i_k}\rangle\, |b_i\rangle,$$

where we assume that the number of records equal to 2^{M+N}. Next we define the oracle function to flag all elements that belong to B_i, the column that we are looking for

$$f(|x\rangle\,|b_j\rangle) = \begin{cases} 1 & \text{ha } j = i \\ 0 & \text{ha } j \neq i. \end{cases}$$

In the example below, there are four columns in the database system encoded as $00, 01, 10$ and 11 where each column has four elements, thus $N = M = 2$ and the total number of records in the table is 16.

As an example a record is referenced as $|01\rangle\,|11\rangle = |0111\rangle$, where $|01\rangle$ is the encoded record and $|11\rangle$ is the corresponding column. The number of iterations needed for an optimal solution in this case is $\left\lfloor \frac{\pi}{4}\sqrt{\frac{16}{4}} \right\rfloor = 1$.

The circuit below implements Grover's algorithm for multiple solution that flags all elements in the 01 column (Fig. 3).

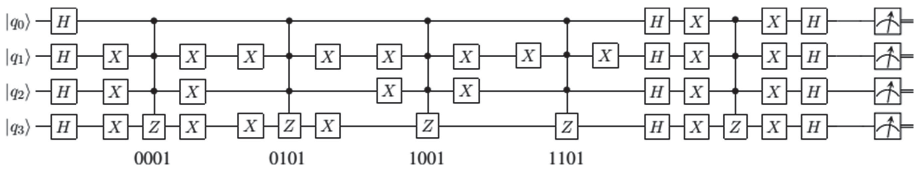


Fig. 3. Making a projection to the column 01 with multiple elements as a solution

8 Evaluation on IBM's Quantum Computers

The proposed algorithms were evaluated and tested on existing quantum computers where the computers' properties such as accuracy and computational time were compared using five publicly available processors. The evaluation was made on IBM's 5-qubit quantum computers where each computer has a different topology and error rate. For reference the algorithms were also tested on the simulator as well which assumes no error rate. The quantum circuits were created using Qiskit which is an open-source quantum computing Python framework.

8.1 Intersection[1]

As seen on Table 2 below, the deployed quantum computers are still working with relatively high error rates (the best result has less than 50% probability).

[1] https://github.com/joczikszabi/ADBIS2020/blob/master/Intersection.ipynb.

Table 2. Results of intersection using IBM quantum computers.

Backend	Result	Computational time	Total runtime
ibmq_simulator	94%	5 ms	15.8 s
ibmq_london	10%	10.3 s	54.1 s
ibmq_burlington	31%	9.6 s	1 m 23.5 s
ibmq_vigo	38%	8.1 s	30 m 45.4 s
ibmqx2	41%	7.4 s	3 m 36 s
ibmq_oursense	44%	8.3 s	3 m 3.1 s

8.2 Set Difference[2]

In case of the set difference operation, we are negating the results from the previous run using X gates in the end. This seems to be increasing the resulting probabilities in almost every case (except on `ibmq_burlington`). An explanation for this could be that adding X gates to the circuit leads to an overall different transpiled circuit which has a lower error rate thus implying a better result (Table 3).

Table 3. Results of difference using IBM quantum computers.

Backend	Result	Computational time	Total runtime
ibmq_simulator	95%	4 ms	29.2 s
ibmq_london	12%	10.3 s	38.5 s
ibmq_burlington	25%	10.5 s	30.5 s
ibmq_vigo	40%	8.5 s	28 m 56.2 s
ibmq_oursense	46%	8.4 s	12 m 25.1 s
ibmqx2	53%	7.5 s	2 m 14.2 s

8.3 Union

Our proposed algorithm for the set union consists of a set difference operation in order to obtain the pairwise disjunct sets $A\backslash B$ and B and the concatenation of these two sets. Therefore the results in this case are equivalent to the ones for the set difference algorithm above.

8.4 Projection[3]

Using the example from the previous section, there were a total of 16 records where 4 of them were marked and searched. Below you can find two tables

[2] https://github.com/joczikszabi/ADBIS2020/blob/master/Difference.ipynb.
[3] https://github.com/joczikszabi/ADBIS2020/blob/master/Projection.ipynb.

where Table 4 describes the computational time as before and Table 5 shows the resulting probabilities for each marked item after the run.

Table 4. Results of projection using IBM quantum computers.

Backend	Result	Computational time	Total runtime
ibmq_simulator	25%	3 ms	4.1 s
ibmq_vigo	5%	9.8 s	43 m 28.1 s
ibmq_burlington	5%	12.4 s	1 m 9.8 s
ibmq_oursense	6%	8.5 s	1 m 58.6 s
ibmqx2	7%	7.7 s	15.4 s
ibmq_london	6%	10 s	54 s

Table 5. Probability results of each marked states

Backend	$\|0001\rangle$	$\|0101\rangle$	$\|0101\rangle$	$\|1101\rangle$
ibmq_simulator	25%	24%	25%	25%
ibmq_vigo	4%	6%	6%	5%
ibmq_burlington	6%	5%	6%	6%
ibmq_oursense	6%	6%	6%	6%
ibmqx2	7%	5%	7%	7%
ibmq_london	6%	5%	8%	7%

As seen above, using Grover's algorithm with multiple marked items leads to lower probabilities than the expected results. It is also notable that the results tend to be around the average on every used backends and none of the computers seem to have exceptionally high probabilities unlike in the previous evaluation.

The implementation follows the one found in the work of Strömberg, P., & Blomkvist Karlsson [10].

9 Conclusion and Future Work

The evaluation of the proposed algorithms in this paper using different IBM-Q Experience platforms allows some statement on how suitable their settings are for the implementation. IBM-Q has five 5-qubit quantum computers which differ in the topology of the qubits and the error rate. Based on the acquired results, quantum computers are still working with relatively high errors and are not efficient for practical use.

Since the resulting probabilities are so low, it'd be challenging to effectively use the proposed algorithms as of now. Reducing the error rates is necessary for the algorithms to properly work and have useful applications in the future. In fact, the presented work has to be seen as an attempt to build a theoretical foundation for future quantum-based database algorithms. Nevertheless improvements can be made for future experimentation. Possible improvements would include creating different circuit designs for the algorithms to reduce the error that is inherited by the different circuits' hardware realization.

The presented set based operations are built upon the intersection operation which exploits the fact that the resulting set is a subset of the power set of the smaller input relation A. The approach is then to enumerate the 2^n sets of the power set and checking all these for containment in the second relation B. Since the number of elements that we need to go through in this case is 2^n, this results in $\mathcal{O}(\sqrt{2^n}) = O(2^{n/2})$ computational time. Therefore using the proposed algorithms in case only a single operation is needed is not suggested since the computational time could be improved using other existing quantum algorithms. But since all proposed operations are inherited from an implementation of set intersection using the well-known Grover's algorithm, this approach would allow for achieving a combined performance gain due to the quantum effects in case of evaluating a combination of multiple set operations as one.

Many different useful operations have been left out of this paper that could be implemented in the future to allow the use of the proposed algorithms for more practical purposes. These operations could include different types of join operations, cross-product and the composition of SQL operations. This implementation then would allow the use of SPJ queries which is crucial for any database systems. However there are many other operations that need to be implemented aswell in order for a database system to work properly such as transactions, concurrence, logging, etc. [11]. Furthermore there are still discussions and significant concerns regarding the applicability of Grover's algorithm for real-life databases whether it could be practical for use or not [12].

References

1. De Wolf, R.: Quantum computing: lecture notes. arXiv preprint arXiv:1907.09415 (2019)
2. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
3. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, pp. 212–219 (1996)
4. Pang, C.Y., Zhou, R.G., Ding, C.B., Hu, B.Q.: Quantum search algorithm for set operation. Quantum Inf. Process. **12**(1), 481–492 (2013)
5. Sakhi, Z., Kabil, R., Tragha, A., Bennai, M.: Quantum cryptography based on Grover's algorithm. In: Second International Conference on the Innovative Computing Technology, pp. 33–37. IEEE, INTECH (2012)
6. Brassard, G., Hoyer, P., Tapp, A.: Quantum algorithm for the collision problem. arXiv preprint quant-ph/9705002 (1997)

7. Lavor, C., Liberti, L., Maculan, N.: Grover's algorithm applied to the molecular distance geometry problem. In: Proceedings of the VII Brazilian Congress of Neural Networks, Natal, Brazil (2005)
8. Baritompa, W.P., Bulger, D.W., Wood, G.R.: Grover's quantum algorithm applied to global optimization. SIAM J. Optim. **15**(4), 1170–1184 (2005)
9. Salman, T., Baram, Y.: Quantum set intersection and its application to associative memory. J. Mach. Learn. Res. **13**, 3177–3206 (2012)
10. Strömberg, P., Blomkvist Karlsson, V.: 4-qubit Grover's algorithm implemented for the ibmqx5 architecture (2018)
11. Roy, S., Kot, L., Koch, C.: Quantum databases. In: Proceedings of the CIDR (No. CONF) (2013)
12. Viamontes, G.F., Markov, I.L., Hayes, J.P.: Is quantum search practical? Comput. Sci. Eng. **7**(3), 62–70 (2005)

Machine Learning

Dynamic k-NN Classification Based on Region Homogeneity

Stefanos Ougiaroglou[1(✉)], Georgios Evangelidis[2], and Konstantinos I. Diamantaras[1]

[1] Department of Information and Electronic Engineering, International Hellenic University, 57400 Sindos, Thessaloniki, Greece
{stoug,kdiamant}@ihu.gr

[2] Department of Applied Informatics, School of Information Sciences, University of Macedonia, 54636 Thessaloniki, Greece
gevan@uom.gr

Abstract. The effectiveness of the k-NN classifier is highly dependent on the value of the parameter k that is chosen in advance and is fixed during classification. Different values are appropriate for different datasets and parameter tuning is usually inevitable. A dataset may include simultaneously well-separated and not well-separated classes as well as noise in certain regions of the metric space. Thus, a different k value should be employed depending on the region where the unclassified instance lies. The paper proposes a new algorithm with five heuristics for dynamic k determination. The heuristics are based on a fast clustering pre-processing procedure that builds an auxiliary data structure. The latter provides information about the region where the unclassified instance lies. The heuristics exploit the information and dynamically determine how many neighbours will be examined. The data structure construction and the heuristics do not involve any input parameters. The proposed heuristics are tested on several datasets. The experimental results illustrate that in many cases they can achieve higher classification accuracy than the k-NN classifier that uses the best tuned k value.

Keywords: k-NN classification · Dynamic k parameter determination · Homogeneous clustering · Heuristics

1 Introduction

The k-NN classifier [4] predicts the class of an instance x by searching in the training set and retrieving the k nearest instances to x. The nearest instances are called neighbours. Then, x is classified to the majority class among the classes that the k nearest neighbours belong to. The majority class is determined via a procedure known as the nearest neighbours voting.

Classification accuracy highly depends on the selection of k. The value of k that achieves the highest accuracy depends on the training set used. Usually, tedious cross-validation tasks are performed to determine the "best" k value. That value is unique and constant for all instances that need to be classified. Although the determination of k can not follow any general rule and the "best" k may be completely different for different training sets, large k values examine larger neighbourhoods and, thus, they have to be used when the classes are not well separated and when the training set contains noise. Therefore, large k values render the classifier more noise tolerant. Small k values render the classifier noise sensitive and should be used on training sets with well-separated classes.

Even the "best" k value can not be optimal. Real-life datasets may have quite different structure in different regions of the metric space. For example, a training set may contain simultaneously well-separated and not well-separated classes as well as noise only in certain regions. In such cases, a classifier that uses a fixed k value may be less accurate than a classifier that utilizes a different k value for each instance that needs to be classified depending on the region where the latter lies. This observation triggered the motivation of the present work.

The contribution of this work is the development of a parameter free k-NN classifier in the sense that it uses a dynamic k value depending on nature of the region where the instance to be classified lies. We call the proposed classifier Region Homogeneity based Dynamic k-NN classifier (rhd-kNN). The rhd-kNN classifier utilizes heuristics that dynamically adjust the k value. The paper introduces five heuristics. All of them are based on a same data structure that is constructed by a fast and parameter-free k-means clustering pre-processing task that builds homogeneous clusters. The data structure holds the cluster centroids as well as information about the area that each cluster centroid represents. In effect, when a new instance x needs to be classified, the nearest centroid c from the data structure is retrieved. Then based on c, k is appropriately adjusted and x is classified by searching the k nearest neighbours in the training set.

Section 2 briefly reviews related work. Section 3 presents in detail the rhd-kNN classifier and the five heuristics. The experimental study is presented in Sect. 4. Section 5 concludes the paper and gives directions for future work.

2 Related Work

In [9] three heuristics for dynamic k value determination are proposed. The three heuristics introduce parameters that should be tuned. In [3] a clustering based method for dynamic k value selection is proposed, but involves various parameters. An interesting proposal to dynamically adjust k is presented in [2]. For each unclassified instance, the algorithm determines the k value by constructing a hypersphere around it to capture the local distribution of the surrounding training instances. In [6], Johansson et al. propose a k-NN classifier that adopts the concept of "Spheres of Confidence" to determine k for each unclassified instance. The work presented in [7] introduced two Adaptive k-NN classifiers. They are code named Ada-kNN1 and Ada-kNN2. Ada-kNN1 uses the density and distribution of the neighborhood of each unclassified instance and learns a suitable k for it by using an artificial neural network. Ada-kNN2 uses a heuristic method guided by an indicator of the local density of the unclassified instance and information about its neighboring training instances.

3 Region Homogeneity Based Dynamic *k*-NN

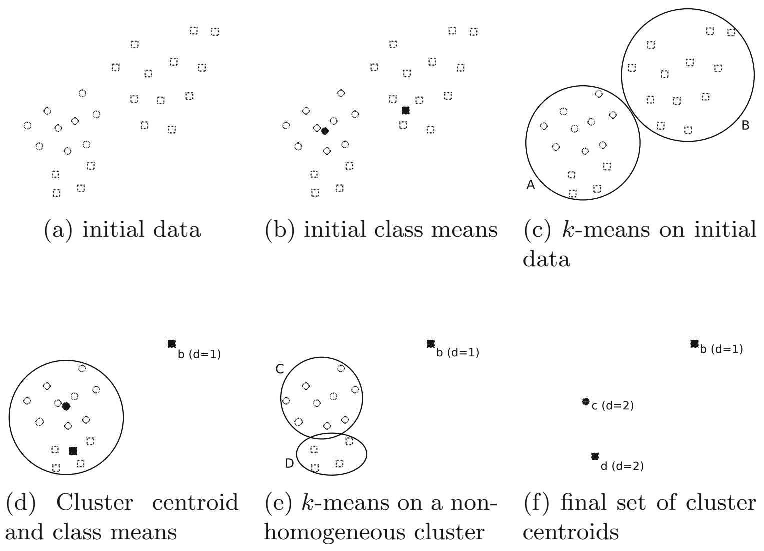
The rhd-kNN classifier is based on a k-means clustering procedure that builds homogeneous clusters and keeps their centroids. The result of the procedure is a data structure, which we call Structure of Homogeneous Clusters (SHC). The concept of homogeneous clustering was first presented in [8] for the purpose of developing a prototype generation data reduction technique. Here, we adopt the same methodology to automatically adjust k for each instance that needs to be classified depending on the nature of the region where the instance lies.

SHC is build by applying the following algorithm: Initially, the training set is considered as a non-homogeneous cluster and a mean instance for each class is computed. Then, k-means is applied using the class means as initial means. The result is the creation of as many clusters as the number of distinct class labels in the cluster. This clustering process is applied recursively for all non-homogeneous clusters, and in the end, all clusters become homogeneous. Each homogeneous cluster centroid is stored in SHC along with a number indicating the recursion depth, i.e., how many recursive calls were necessary to determine that homogeneous cluster.

Figure 1 presents an example. Assume that the training set has 26 instances that can be either “squares” or “circles”. The SHC construction algorithm computes a mean for the class “square” and a mean for the class “circle” (see Fig. 1(b)). Then, k-means is executed by using the class means as initial means and produces two clusters (Fig. 1(c)). Cluster A is non-homogeneous and cluster B is homogeneous. The algorithm stores the centroid of cluster B to SHC along with the number $d = 1$, denoting that the homogeneous cluster was produced at recursion depth 1. For cluster A, the class means in the cluster are computed (Fig. 1(d)), k-means is executed and discovers clusters C and D. Both are homogeneous and their centroids are placed in SHC along with the number $d = 2$ since both were produced at recursion depth 2 (Fig. 1(f)).

The aforementioned example can be illustrated as a tree of clusters (see Fig. 2). The root is the whole training set. The first level of the tree holds clusters A and B. Since cluster B is homogeneous, it becomes a leaf. Cluster A becomes parent of clusters C and D. Since C and D are homogeneous, they also become leaves. Obviously, for large regions that include instances of only one class label, the SHC construction algorithm discovers large homogeneous clusters at a relatively low recursion depth. These clusters are leaves and are placed not far away from the root. For close class border or noisy regions, the algorithm identifies small homogeneous clusters at higher recursion depths. Those clusters are placed far away from the root. Hence, the cluster centroids hold information about the region they represent in the form of the recursion depth d.

The rhd-kNN classifier utilizes SHC to determine the k value to be used for each individual instance that needs to be classified. The SHC construction



(a) initial data (b) initial class means (c) k-means on initial data

(d) Cluster centroid and class means (e) k-means on a non-homogeneous cluster (f) final set of cluster centroids

Fig. 1. Data generation through recursive k-means clustering

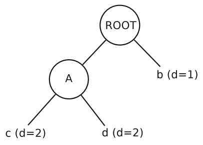


Fig. 2. Tree representation of SHC

algorithm runs only once as a pre-processing step. Then, when a new instance x needs to be classified, rhd-kNN finds the 1-nearest centroid c in SHC and its corresponding d. Then, one of the heuristics is employed to determine k based on d. Finally, rhd-kNN classifies x by finding the k nearest neighbours in the training set. The five proposed heuristics are summarized below:

- $k = d$: It just defines k to be equal to the depth of the 1-nearest centroid in SHC. This heuristic is used as a baseline since often a larger number of neighbours is more appropriate.
- $k = 2^d$: This heuristic tends to examine an extremely large number of nearest neighbours, especially when d is greater than 9. Thus, in our experiments, we manually set $k = 2^9$ when $d > 9$.
- $k = d^2$: This heuristic is a trade-off between the above two heuristics.
- $k = (d \times (d + 1))/2$ or $k = \sum_{i=1}^{d} d$: This heuristic determines k by mapping values of d to the following arithmetic sequence: $1, 3, 6, 10, 15, 21, 28, \ldots$.
- $k = \lfloor e^{\sqrt{d}} \rfloor$: This is a more conservative heuristic than the previous one. It uses the $2, 4, 5, 7, 9, 11, 14, \ldots$ sequence for determining k.

We expect that our classifier will outperform the k-NN classifiers that use fixed k parameter values for datasets that contain a mixture of well-separated and not well-separated classes, like the well-known iris dataset. The SHC construction algorithm will build only one homogeneous cluster for the region containing the well-separated class (in our case iris-setosa). Thus, the proposed heuristics will consider a very small k value for each instance lying closer to that cluster centroid. For unclassified instances close to cluster centroids in the regions of the other two classes the algorithm will use larger k values.

4 Performance Evaluation

4.1 Experimental Setup

The rhd-kNN classifier was evaluated on fourteen datasets. Table 1 summarizes their characteristics. They are distributed by the KEEL dataset repository[1] [1]. We used the Euclidean distance as the distance metric. The datasets were normalized within the range $[0, 1]$.

We wanted to test the performance of rhd-kNN on datasets with noise. Thus, for some of the datasets, we built two additional "noisy" versions by adding 10% and 30% random uniform noise. The noise was added by setting the class label of the 10% or 30% of the instances to a randomly chosen different class label. The datasets on which we artificially added noise are code-named by their abbreviation plus the level of added noise (e.g., txr30).

We compared the performance of rhd-kNN against the conventional k-NN classifiers that use fixed k values. We divided each dataset into a training set and a testing set. We added noise only in the training portions. Possible ties during the majority class voting were resolved using the 1-nearest neighbour rule. We built six conventional k-NN classifiers with constant k parameter value. The first three classifiers are: the widely used 1-NN classifier, the 5-NN classifier (5 is the default value for the implementation of k-NN classifier in Python's scikit-learn library) and the 10-NN classifier. The fourth and fifth conventional k-NN classifiers used are those with $k = \sqrt{N}$ [2,5] and $k = \sqrt{\frac{N}{2}}$ where N is the number of instances in the training set. They are common rule-of-thumb (RoT) approaches that are often utilized in the literature. The last conventional k-NN classifier used is that with the "best" k parameter value.

"Best k" was estimated by applying a 5-fold cross validation schema. We divided each training set into five portions. Then, we ran the k-NN classifier five times. Each time, a different portion was the validation set. Each instance of the validation set was classified by applying the k-NN classifier that searches for nearest neighbours into the union of the rest four portions. The result was the average accuracy of the five executions. We applied the aforementioned procedure fifty times by varying k from 1 to 50. Then, we kept the k parameter value that achieved the highest accuracy, and this is the so called "best" k. Obviously,

[1] http://sci2s.ugr.es/keel/datasets.php.

Table 1. Dataset description

Dataset	Size	Attributes	Classes
Balance (bl)	625	4	3
Banana (bn)	5300	2	2
Ecoli (ecl)	336	7	8
Iris	120	4	3
Letter Recognition (lir)	20000	16	26
Landsat Satellite (ls)	6435	36	6
Magic G. Telescope (mgt)	19020	10	2
Pen-Digits (pd)	10992	16	10
Phoneme (ph)	5404	5	2
Pima (pm)	615	8	2
Shuttle (sh)	58000	9	7
Twonorm (tn)	7400	20	2
Texture (txr)	5500	40	11
Yeast (ys)	1484	8	10

the accuracy achieved by the k-NN classifier that uses the "best" k is derived by classifying the instances of the initial testing set. For the datasets with artificially added noise, we estimated the "best" k value by using validation sets without noise. To achieve this, the noise was added in a copy of the original training set. Then, the original training set and the "noisy" copy were divided into five folds. Then, we kept the five "noisy" training sets from the "noisy" copy and the five validation sets from the original training set.

The Nearest cluster centroid classifier that assigns an instance to the class of the nearest cluster centroid in SHC is identical to RHC [8]. In effect, RHC generates the cluster centroids (prototypes) that then are used as training data for the k-NN classifier. Like other condensing and prototype generation algorithms, RHC aims to reduce the data as much as possible without significant loss of accuracy. As presented in [8], RHC and other relevant algorithms cannot achieve higher accuracy than the conventional k-NN classifier. Therefore, we did not include RHC in the present experimental study.

We conducted the experiments without prior knowledge about the datasets. We believe that rhd-kNN could be more accurate than the "best" k-NN classifier only when the datasets include simultaneously well-separated and not well-separated classes as well as noise in certain regions. However, we conducted experiments by using datasets that may not belong to such dataset categories.

Apart from the accuracy, we estimated three computational cost measurements in terms of distance computations. The first one concerns the cost of the k-NN classifier. The k value does not influence that cost when brute force is used to retrieve the nearest neighbours. Thus, rhd-kNN and conventional k-NN classifiers need to compute that number of distances. The second measurement concerns the cost overhead for searching for the nearest cluster centroid in SHC and it concerns exclusively rhd-kNN. The last cost measurement is the pre-processing cost required for the SHC construction.

4.2 Experimental Results

Table 2 presents the computational cost measurements. The last column lists the number of distances computed for SHC construction, which obviously is a computationally "cheap" algorithm. Considering that the SHC construction algorithm runs only once as a pre-processing step, the computational cost is insignificant. The computational cost of cross-validation needed for parameter tuning in the case of "best k" is not reported. Bear in mind that it is considerably higher than the computational cost of the SHC construction algorithm.

The other two columns present the distance computations for the classification step. All the classifiers of the experimental study have to compute the distances listed in column "NN search over TS". The rhd-kNN classifiers, in addition, have to compute the distances that concern the search of the nearest cluster centroid in SHC, listed in column "NN search over SHC".

Table 3 presents the accuracy measurements. Almost in all cases an rhd-kNN classifier can achieve higher accuracy than the accuracy achieved by the conventional k-NN classifiers with $k = 1$, $k = 5$, $k = 10$, $k = \sqrt{N}$ and $k = \sqrt{\frac{N}{2}}$. The 10-NN performs quite well on the specific suite of datasets, since a large number of them contain noise. Notice, though, that rhd-kNN classifiers in many cases clearly beat all versions of kNN with fixed k (excluding *best* k-NN). This is demonstrated in the bl, bl10, ecl30, iris, ph30, pm10, txr30 and ys datasets. Finally, rhd-kNN classifiers almost always outperform RoT classifiers. This is the reason that in Table 3 we indicate with boldface only the winners among *best* k-NN and the rhd-kNN classifiers.

The comparison between the *best* k-NN and rhd-kNN reveals noteworthy performance for rhd-kNN. At least one of the rhd-kNN classifiers can achieve higher accuracy than that of the *best* k-NN classifier in 18 datasets, while in two datasets, a rhd-kNN approach is as accurate as the *best* k-NN classifier. Contrary to the *best* k-NN classifier, rhd-kNN achieves that performance without the need of any input parameter and tedious and costly parameter tuning procedures.

The $k = (d \times (d + 1))/2$ heuristic seems to be an ideal approach since it achieves high accuracy even when the dataset contains noise. In nine datasets it is more accurate than *best* k-NN classifier. The simple $k = d$ heuristic performs well on datasets that in their original form do not include noise (e.g., lir, pd, sh, txr). The $k = 2^d$, $k = d^2$ and $k = \lfloor e^{\sqrt{d}} \rfloor$ heuristics all achieve accuracies that are close to those of the *best* k-NN classifier and, in some cases, even better.

The comparison between shd-kNN and *best* k-NN on "noisy" versions of the datasets does not reveal any useful insights on which classifier is more accurate on "noisy" conditions. The noise leads to smaller clusters with high d values.

Table 2. Computational cost in terms of distance computations

Dataset	NN search over TS	NN search over SHC	Construction of SHC
bl	62,500	12,500	59,159
bl10	62,500	21,125	45,104
bl30	62,500	30,250	52,568
bn	4,494,400	940,220	592,642
ecl	18,023	6,030	41,592
ecl10	18,023	6,901	38,263
ecl30	18,023	12,596	43,200
iris	3,600	390	3,826
lir	64,000,000	7,476,000	37,168,151
ls	6,625,476	679,536	1,751,252
ls10	6,625,476	1,537,965	1,945,468
ls30	6,625,476	2,626,767	1,968,248
mgt	57,881,664	12,035,856	3,830,966
mgt10	57,881,664	17,296,788	3,918,354
pd	19,329,212	655,004	2,593,601
pd10	19,329,212	5,303,774	4,231,769
pd30	19,329,212	10,132,780	4,298,161
ph	4,669,920	886,680	639,564
ph10	4,669,920	1,710,720	693,068
ph30	4,669,920	2,494,800	766,212
pm	94,095	27,846	59,688
pm10	94,095	35,343	58,802
pm30	94,095	36,261	65,418
sh	538,193,600	1,774,647	10,977,178
tn	8,761,600	307,840	1,564,256
tn10	8,761,600	1,147,000	1,590,280
tn30	8,761,600	1,863,320	1,793,078
txr	4,840,000	257,400	2,623,438
txr10	4,840,000	1,250,700	3,403,100
txr30	4,840,000	2,523,400	3,521,357
ys	351,648	176,712	431,125
ys10	351,648	204,832	581,014
ys30	351,648	256,336	390,432

Thus, all heuristics use a high k. Accordingly, the cross-validation tuning process reveals a high k value for the *best* k-NN classifier.

Table 3. k-NN classification with fixed and dynamic k parameter values

Data	Best	Best	1-NN	5-NN	10-NN	RoT	RoT	rhd-kNN				
	k value	k-NN				$k = \sqrt{N}$	$k = \sqrt{\frac{N}{2}}$	$k = d$	$k = 2^d$	$k = d^2$	$k = (d\times(d+1))/2$	$k = \lfloor e^{\sqrt{d}} \rfloor$
bl	42	89.60	79.20	86.40	89.60	89.60	89.60	84.80	88.80	89.60	**90.40**	89.60
bl10	43	88.80	68.80	86.40	87.20	89.60	89.60	88.00	**91.20**	89.60	90.40	87.20
bl30	32	**88.00**	59.20	69.60	79.20	83.20	83.20	73.60	83.20	87.20	82.40	76.80
bn	29	**90.66**	87.26	89.81	90.19	90.57	90.47	89.81	88.68	90.38	90.28	90.47
ecl	6	**91.05**	83.58	89.55	92.54	86.57	92.54	88.06	88.06	86.57	86.57	89.55
ecl10	6	89.55	77.61	88.06	92.54	88.06	89.55	86.57	91.05	**92.54**	91.05	91.05
ecl30	14	85.08	61.19	82.09	85.08	85.08	85.08	76.12	85.08	**88.06**	**88.06**	85.08
iris	11	93.33	90.00	93.33	93.33	93.33	93.33	93.33	93.33	93.33	**96.67**	93.33
lir	4	**95.78**	95.70	95.40	95.03	81.05	84.20	95.65	95.28	93.83	95.05	95.43
ls	8	**91.53**	89.98	91.53	91.14	86.09	87.34	91.22	90.68	89.59	90.99	90.83
ls10	8	**91.30**	81.66	90.52	90.99	86.09	87.10	90.75	90.37	89.90	89.98	90.75
ls30	13	88.27	61.23	82.21	89.04	86.25	86.79	82.98	88.58	89.36	**88.81**	87.72
mgt	10	**83.57**	80.13	82.97	83.57	80.97	81.65	83.36	78.97	80.86	81.86	83.39
mgt10	20	**83.10**	73.32	80.97	82.76	81.13	81.55	82.89	78.68	80.81	81.84	82.99
pd	1	99.05	99.05	99.09	98.73	95.04	96.13	**99.09**	98.59	98.32	98.54	98.91
pd10	8	98.86	89.40	98.95	98.73	95.13	96.36	**98.91**	98.73	98.36	98.64	**98.91**
pd30	12	**98.68**	69.75	94.18	98.45	94.90	96.36	92.58	98.04	98.23	98.41	97.59
ph	1	**88.70**	88.70	86.94	86.30	82.04	83.52	85.74	78.98	81.30	81.94	84.44
ph10	8	**86.67**	81.39	86.02	86.02	81.76	83.89	84.35	78.80	81.11	81.48	84.26
ph30	48	80.19	65.46	72.32	75.28	80.93	80.65	77.96	79.07	80.28	**80.93**	79.44
pm	48	76.47	77.12	78.43	81.05	75.16	77.78	80.39	72.55	74.51	73.86	**82.35**
pm10	18	78.43	72.55	76.47	78.43	75.16	78.43	75.82	77.12	77.78	**79.09**	77.78
pm30	47	73.86	64.05	64.71	69.94	71.90	71.24	71.24	73.86	71.90	73.20	**76.47**
sh	1	**99.96**	99.96	99.91	99.86	99.44	99.36	99.89	99.72	99.67	99.83	99.85
tn	49	97.97	95.54	97.70	97.77	97.77	98.04	97.77	**98.11**	97.70	97.91	98.04
tn10	47	97.77	84.39	95.47	97.10	97.57	97.50	97.10	**97.91**	97.43	97.70	97.70
tn30	50	**97.77**	67.23	79.19	86.08	97.43	97.57	86.69	**97.77**	97.03	97.57	94.73
txr	1	**98.73**	98.73	98.09	97.82	94.73	95.82	98.36	98.09	97.55	98.00	98.09
txr10	5	97.91	90.55	97.91	97.82	94.55	95.55	**98.09**	97.73	97.46	97.64	97.73
txr30	10	**97.55**	69.73	93.55	97.55	94.82	94.91	91.09	96.55	97.00	**97.55**	96.64
ys	14	58.45	49.32	57.10	59.12	61.15	60.14	55.41	61.15	60.14	**61.49**	57.43
ys10	13	58.45	44.93	52.37	58.11	60.14	58.11	51.01	59.46	**59.80**	57.43	57.43
ys30	14	56.76	36.82	45.61	55.41	58.78	59.46	44.26	50.00	**57.77**	56.08	48.99
AVG:	19.42	**87.63**	76.77	84.33	86.72	85.33	86.15	84.63	86.19	86.82	87.02	86.70

Table 4 gives an insight on the determination of k based on d. For each dataset, we report the number of testing instances whose 1-nearest cluster centroid in SHC is at a given recursion depth. In general the number of instances follow the normal distribution for almost all datasets. There are some exceptions, like the ph dataset, where a large number of instances are assigned to a small depth. Also, in many datasets the distribution is left-skewed. As expected, we observe shifted to the right distributions (larger d values) for noisy datasets.

Table 4. Number of instances whose 1-nearest cluster centroid in SHC was at depth i

Dataset	Recursion depth of the 1-nearest cluster centroid																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
bl	0	11	14	27	61	12	0	0	0	0	0	0	0	0	0	0	0	0
bl10	0	0	1	22	49	46	6	1	0	0	0	0	0	0	0	0	0	0
bl30	0	0	0	18	70	33	3	1	0	0	0	0	0	0	0	0	0	0
bn	0	0	0	0	42	143	124	146	110	140	183	120	51	1	0	0	0	0
ecl	0	18	11	17	14	6	1	0	0	0	0	0	0	0	0	0	0	0
ecl10	0	10	31	19	7	0	0	0	0	0	0	0	0	0	0	0	0	0
ecl30	0	8	27	24	7	1	0	0	0	0	0	0	0	0	0	0	0	0
iris	10	7	6	2	5	0	0	0	0	0	0	0	0	0	0	0	0	0
lir	101	737	1595	1157	343	62	5	0	0	0	0	0	0	0	0	0	0	0
ls	98	103	53	338	229	235	189	41	1	0	0	0	0	0	0	0	0	0
ls10	0	4	26	362	656	226	13	0	0	0	0	0	0	0	0	0	0	0
ls30	0	0	5	229	768	274	11	0	0	0	0	0	0	0	0	0	0	0
mgt	0	0	0	0	29	52	46	74	108	321	600	915	926	509	175	48	1	0
mgt10	0	0	0	0	0	0	5	21	54	169	437	870	1016	680	388	132	30	2
pd	303	706	713	343	120	13	0	0	0	0	0	0	0	0	0	0	0	0
pd10	0	0	180	888	794	273	59	3	1	0	0	0	0	0	0	0	0	0
pd30	0	0	58	903	1036	180	20	1	0	0	0	0	0	0	0	0	0	0
ph	0	106	29	37	69	217	163	156	167	99	32	1	4	0	0	0	0	0
ph10	0	0	0	0	0	2	17	26	93	169	267	206	190	61	42	1	6	0
ph30	0	0	0	0	0		2	13	79	132	220	262	192	119	49	12	0	0
pm	0	0	0	0	0	23	44	41	38	6	1	0	0	0	0	0	0	0
pm10	0	0	0	0	4	8	25	45	43	12	16	0	0	0	0	0	0	0
pm30	0	0	0	0	3	8	27	34	29	40	12	0	0	0	0	0	0	0
sh	0	1808	2390	2998	1223	1673	605	613	197	69	23	0	0	0	0	0	0	0
tn	0	0	0	74	130	203	281	315	337	114	26	0	0	0	0	0	0	0
tn10	0	0	0	0	0	0	33	133	476	595	211	32	0	0	0	0	0	0
tn30	0	0	0	0	0	0	0	4	152	661	572	88	3	0	0	0	0	0
txr	168	170	331	341	75	15	0	0	0	0	0	0	0	0	0	0	0	0
txr10	0	7	256	556	244	34	3	0	0	0	0	0	0	0	0	0	0	0
txr30	0	1	136	713	221	27	2	0	0	0	0	0	0	0	0	0	0	0
ys	0	6	18	91	133	36	7	5	0	0	0	0	0	0	0	0	0	0
ys10	0	5	45	137	76	29	4	0	0	0	0	0	0	0	0	0	0	0
ys30	0	3	74	168	49	2	0	0	0	0	0	0	0	0	0	0	0	0

5 Conclusions

We propose the rhd-kNN classifier, a parameter free k-NN classifier that heuristically determines how many neighbours will be examined for each unclassified instance. Hence, a different k value is employed depending on the region where the instance lies. The classifier uses a pre-processing step that builds an auxiliary data structure (SHC). The latter holds the centroids of homogeneous clusters obtained via a k-Means clustering task together with their depth (a number indicating the recursion depth when the clusters were formed). When an instance needs to be classified, SHC provides information about the region where the instance lies, in effect, whether the instance is in a noisy region in terms of class labels or not. Then the heuristic used exploits the information and dynamically determines how many neighbours will be examined. The proposed heuristics are tested on several datasets. The results show that in many cases they can achieve higher accuracy than the k-NN classifier that uses the best tuned k value.

We plan to further explore dynamic k parameter determination. We plan to develop heuristics that take into consideration additional metrics about the region such as the number of instances the centroid represents and the number of distinct class labels that exist in the immediate neighborhood of the instance.

References

1. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Mult. Valued Log. Soft Comput. **17**(2–3), 255–287 (2011)
2. Bhattacharya, G., Ghosh, K., Chowdhury, A.S.: Test point specific k estimation for kNN classifier. In: Proceedings of the 2014 22nd International Conference on Pattern Recognition, ICPR 2014, USA, pp. 1478–1483. IEEE Computer Society (2014). https://doi.org/10.1109/ICPR.2014.263
3. Bulut, F., Amasyali, M.F.: Locally adaptive k parameter selection for nearest neighbor classifier: one nearest cluster. Pattern Anal. Appl. **20**(2), 415–425 (2015). https://doi.org/10.1007/s10044-015-0504-0
4. Dasarathy, B.V.: Nearest neighbor. NN pattern classification techniques. IEEE Computer Society Press, NN norms (1991)
5. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, Hoboken (2000)
6. Johansson, U., Boström, H., König, R.: Extending nearest neighbor classification with spheres of confidence. In: Wilson, D., Lane, H.C. (eds.) Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference, Coconut Grove, Florida, USA, 15–17 May 2008, pp. 282–287. AAAI Press (2008). http://www.aaai.org/Library/FLAIRS/2008/flairs08-070.php
7. Mullick, S.S., Datta, S., Das, S.: Adaptive learning-based k -nearest neighbor classifiers with resilience to class imbalance. IEEE Trans. Neural Networks Learn. Syst. **29**(11), 5713–5725 (2018). https://doi.org/10.1109/TNNLS.2018.2812279
8. Ougiaroglou, S., Evangelidis, G.: RHC: a non-parametric cluster-based data reduction for efficient k-NN classification. Pattern Anal. Appl. **19**(1), 93–109 (2014). https://doi.org/10.1007/s10044-014-0393-7
9. Ougiaroglou, S., Nanopoulos, A., Papadopoulos, A.N., Manolopoulos, Y., Welzer-Druzovec, T.: Adaptive k-nearest-neighbor classification using a dynamic number of nearest neighbors. In: Ioannidis, Y., Novikov, B., Rachev, B. (eds.) ADBIS 2007. LNCS, vol. 4690, pp. 66–82. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75185-4_7

Contextualisation of Datasets for Better Classification Models: Application to Airbus Helicopters Flight Data

Marie Le Guilly[1(✉)], Nassia Daouayry[1,2], Pierre-Loic Maisonneuve[2], Ammar Mechouche[2], Jean-Marc Petit[1], and Vasile-Marian Scuturici[1]

[1] Univ Lyon, INSA Lyon, LIRIS (UMR 5205 CNRS), Villeurbanne, France
{marie.le-guilly,jean-marc.petit,marian.scuturici}@insa-lyon.fr
[2] Airbus Helicopters, Marignane, France
{nassia.daouayry,pierre-loic.maisonneuve,ammar.mechouche}@airbus.com

Abstract. For helicopters, anticipating failures is crucial. To this end, the analysis of flight data allows to develop predictive maintenance approaches, for which Airbus Helicopters (AH) has proposed several solutions, some based on machine learning using predictive models. One recurrent problem in this setting is the *contextualization* of the data, that is to identify the data better fitting the phenomenon being modeled. Indeed, helicopters are complex systems going through different flight phases. Experts therefore have to identify the adequate ones, in which the selected flight parameters are stable and consistent with the studied problem. In this paper, we propose a generic solution to contextualize classification data, and present an experimental study on AH flight data: the results are encouraging and allow to keep domain experts involved the process.

Keywords: Data contextualization · Failure anticipation · Classification

1 Introduction

In the helicopters industry, predictive maintenance is crucial and Airbus Helicopters (AH) seeks to anticipate failure as soon as possible. One solution is to analyze flight data, as most helicopters are equipped with flight recorders for hundreds of parameters. Such an amount of data makes it possible to analyse "low-level signals" over longer periods of time, and to detect failures earlier. In this context, AH has gathered data on hundreds of thousands flight hours: to face such a huge amount of data, a Big Data platform has been deployed at AH to enable the storing and processing of large quantities of data [9].

Using this platform, *digital twins* have been devised to identify as soon as possible small variations on core physical sensors. They are mainly based on physical models and expert knowledge, but AH combines these with machine

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 38–48, 2020.
https://doi.org/10.1007/978-3-030-54623-6_4

learning techniques to build predictive models from the data. To build such models, AH faces generic and recurrent issues that are well-known in machine learning, such as data cleaning, accuracy, or explainability. But in addition to these classic issues, AH also seeks to build models corresponding to the *normal behavior* of the system, and has to use data fitting the behavior algorithms have to model. Indeed, an important filtering step is performed to identify the data that is adequate to deal with the considered problem: complex systems such as helicopters go through many different phases, and only a subset of the data is relevant for a given model, as they are the only one for which the laws of normal behavior of the system apply. It is therefore necessary to identify the correct context for the considered task, which is the subset of data corresponding to the desired phases on which the model is applied. We define this problem as *contextualization*, according to the term used by AH experts. Thus, it consists in determining the flight phases where considered parameters have lesser variability and are less subject to pilot maneuvers and external parameters not recorded by the system. At AH, this crucial step is dealt with by relying on experts knowledge who specify how to filter flight data.

The contextualization problem can seem as a simple problem at first hand (mainly data selection), but turns out to be a nightmare in practice. Identifying the appropriate data is clearly not an easy task, and depends on the final objective for the classification model. In addition, contexts are tightly linked with the application they concern, so solutions are often specific to one given situation. For systems such as helicopters, contextualization is also important as they are systems governed by physical laws, that apply only in specific contexts: the purpose of classification models is therefore to produce outputs coherent with these laws. To this end, these models have to be trained on data consistent with the physical model they represent.

In this paper, we propose our ongoing work to address the contextualization challenge. We seek to identify the appropriate context for a classification task, by identifying the subset of data more likely to capture the normal behavior. To do so, we seek the data favoring the existence of a function between the features and the class to predict. As the correct context should follow some underlying function the model seeks to define, we propose to remove the regions of the data preventing the existence of that function, and to only keep the data more likely to correspond to a normal behavior. We then show how this approach can be applied to AH classification datasets.

Based on these considerations, we made the following contributions: (1) Proposing a generic solution for contextualization, in order to define filters that can be used to reduce the dataset to a given context; (2) Experiments on AH data showing how identifying context elements can improve the accuracy of classifiers; (3) Confronting a contextualization proposed by AH experts to additional context elements proposed by our method.

Section 2 introduces the preliminaries. In Sect. 3, we propose our approach to better contextualize datasets, and in Sect. 4, we focus on AH's data, to show how we built a context for the considered dataset, and develop the lessons drawn from this collaboration based on the experimentations that have been conducted. Finally Sect. 5 presents the related work before concluding in Sect. 6.

2 Preliminaries

2.1 Functional Dependencies

We first recall basic notations and definitions (see [8]). Let U be a set of attributes. A relation schema R is a name associated with attributes of U, i.e. $R \subseteq U$. A database schema $\mathcal{R}$ is a set of relation schemas. Let D be a set of constants, $A \in U$ and R a relation schema. The domain of A is denoted by $dom(A) \subseteq D$. A tuple t over R is a function from R to D. A relation r over R is a set of tuples over R. If $X \subseteq U$, and if t is a tuple over U, then we denote the restriction of t to X by $t[X]$. If r is a relation over U, then $r[X] = \{t[X], t \in R\}$.

Definition 1. *Let R be a relation schema, $X \subseteq R$ and $C \subseteq R \backslash X$. A FD on R is an expression of the form $R : X \rightarrow C$ (or simply $X \rightarrow C$ when R is clear from context)*

Definition 2. *Let r be a relation over R and $X \rightarrow C$ a functional dependency on R. $X \rightarrow C$ is satisfied in r, denoted by $r \models X \rightarrow C$, if and only if for all $t_1, t_2 \in r$, if $t_1[X] = t_2[X]$ then $t_1[C] = t_2[C]$.*

2.2 Supervised Classification in Machine Learning

Let's consider a set of N training samples $\{(x_1, y_1), ..., (x_N, y_N)\}$ where x_i is the feature vector of the i-th example and y_i its label (or class). The number of different labels K, is limited and much smaller than the number of samples. Given this, classification is the task of learning a target function g (a classifier) that maps each example x to one of the k classes, with the lowest error rate. It is possible to express a classification problem using relational databases notations. In the sequel, we will therefore consider a relation $r_0(A_1, \ldots, A_n, C)$ with N tuples, where for any tuple t_i, $t_i[A_1 \ldots A_n] = x_i$ and $t_i[C] = y_i$. In addition, we consider that traditional feature selection methods (see [1]) have been applied and consider the subset $X \subseteq \{A_1 \ldots A_n\}$ of selected features.

To evaluate the performances of an algorithm, we use accuracy, which is the proportion of samples that are correctly classified by a model. This score lies between 0 and 1, and ideally should get as close as possible to 1. Given a model M over a relation r, accuracy is defined as follows:

$$accuracy(M, r) = \frac{\#\, of\, correct\, predictions}{|r|}$$

2.3 Existence Versus Determination of a Function

We use the link between FDs and classification, developed in [7]. We only underline here it relies on the notion of function, as classifier seeks to define a function from the features to the class, while the FD $X \rightarrow C$ can say whether or not such a function exists or not: the FD $X \rightarrow C$ is satisfied if and only if there exists a function from X to C. If the FD is not satisfied, it means some pairs of tuples have the same value on X, but different classes. Such tuples are called counterexamples:

Definition 3. *Let r be a relation over R and $X \rightarrow C$ a FD f on R. The set of counterexamples of f over r is denoted by $CE(X \rightarrow C)$ and defined as follows:*

$$CE(X \rightarrow C, r) = \{(t_1, t_2) | t_1, t_2 \in r, t_1[X] = t_2[X] \; and \; t_1[C] \neq t_2[C]\}$$

Counterexamples are important as they identify pairs of tuples for which the classifier cannot perform correctly, as for the same input, it always predicts the same output. The proportion of counterexamples therefore directly impacts the quality of the classification: it can be evaluated using measure G_3, and contrary to [5] that presents this measure as an error, we propose it as follows:

$$G_3(X \rightarrow C, r) = \frac{max(\{|s| | s \subseteq r, s \models X \rightarrow C\})}{|r|}$$

Measure G_3 is of crucial importance for the classification problem, as in the subset s defined for G_3, there exists a function between the left and right hand side of the dependency. For classification, measure G_3 is therefore a way to bound the accuracy a classifier can reach on the considered dataset, as it is necessary limited by the existence of counterexamples. As a result, the following result holds, for which the details and proof are given in [7]:

Proposition 1. *Let $X \subseteq R$ be a set of features, $C \in R$ the class to be predicted, r a relation over R, and M a classifier from X to C. Then:*

$$accuracy(M, r) \leq G_3(X \rightarrow C, r)$$

In the setting of contextualization, G_3 can be seen as a way to identify whether or not a dataset follows a function, and to identify zones that are therefore more likely to correspond to a normal behavior of the system.

3 Contextualization of a Classification Dataset

The objective is to propose a methodology for the contextualization of classification datasets. The proposed solution considers there should be a function between the features and the class to predict. The idea is to identify the regions in the initial dataset in which a function is likely to exist, and therefore in which the FD $features \rightarrow class$ is likely to be satisfied. On the opposite, regions with a high proportion of counterexamples should be removed, as they are likely regions where the model hypothesis are not verified.

To contextualize a dataset, we propose an iterative approach, that is summarized on Fig. 1. The process starts with an initial classification dataset. It is then discretized, to smooth the data variability and to better identify counterexamples. Then, G_3 is computed, and a classifier is trained and tested, to obtain an accuracy measure. Measure G_3 allows to evaluate the existence of a function, while the accuracy guaranties the performances of the model. These two measures are taken into account to determine the next step in the process. If the domain experts are not satisfied with the measures, the counterexamples are enumerated, to identify filters to remove the tuples that cause too many counterexamples. The key is to find balance between removing regions of the data while keeping as many tuples as possible. The filters can take different forms: here, we propose to define filters in the form of conjunctions of conditions allowing to remove groups of tuples. To identify such groups, visualizations are proposed, to observe what tuples are the most involved in counterexamples. Once the filters are determined, based on these visualizations and in collaboration with domain experts, tuples are removed, providing a new dataset. This process is repeated until satisfaction.

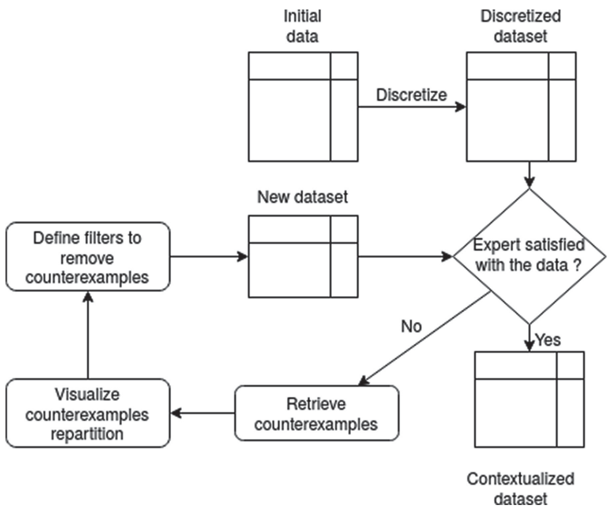
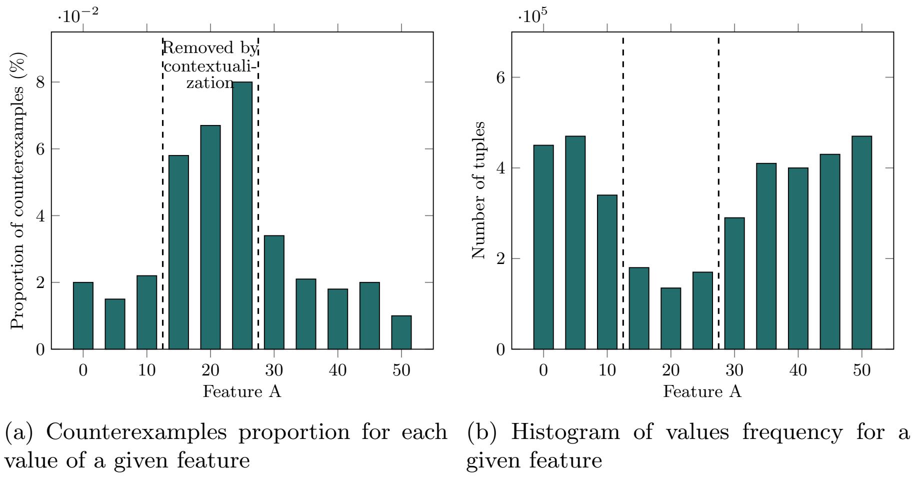

Initial data
Discretized dataset
Discretize
New dataset
Define filters to remove counterexamples
Expert satisfied with the data ?
No
Yes
Visualize counterexamples repartition
Retrieve counterexamples
Contextualized dataset

Fig. 1. Overview of the solution proposed to contextualize a classification dataset

3.1 From Counterexamples to Context-Aware Data Selection

When the proposed contextualization is not satisfying, solutions have to be proposed to refine it, and to therefore remove tuples from the dataset. The challenge is to determine what are the tuples to remove and why. We therefore propose to determine filters that can be applied to the dataset, to remove tuples and lower the number of counterexamples in the dataset. Such filter should ideally remove as few tuples as possible, while removing as many counterexamples as possible. Indeed, one tuple might be involved in many counterexamples: in this case, it should be removed.

Many solutions can be considered for the filters: one solution from example is too order the tuples by the number of counterexamples they are involved in, and to set a threshold to remove all the tuples involved in more counterexamples than this threshold. But it does not explain what are the characteristics of the removed tuples: if a domain expert wishes to understand why a tuples is removed, she has to manually check each counterexample. In this paper, we propose to define filters in the form of conjunction of conditions applied to the dataset, making the overall process explainable. These filters define, in simple terms, regions of the dataset containing more counterexamples than others, while concerning only a few tuples. This can be performed using visualizations proposing, for each feature, histograms showing the distribution of values among counterexamples, and the number of tuples taking a given value. The histograms can then be used to identify values having, on a given feature, few tuples involved in many counterexamples. The filters then integrate a condition removing such values from the dataset. Such filters are interpretable by domain experts, who can analyze whether or not these filters make sense with the desired context.



(a) Counterexamples proportion for each value of a given feature

(b) Histogram of values frequency for a given feature

Fig. 2. Toy example for filter design

Example 1. Figure 2 presents visualizations used to define filters. For a given feature A, Fig. 2a shows for each value taken by this feature, the proportion of tuples involved in counterexamples, and therefore how much they contribute to the value of G_3. Figure 2b is an histogram of values for the considered feature. By comparing these two visualizations, it appears there is a zone that does not contain many tuples, but many counterexamples. As a result, one condition for a contextualization filter could be to remove all tuples for which $A \geq 15$ and $A \leq 25$. This gives an interpretable filter, removing a few tuples and improving measure G_3. Similar work can be performed for each feature of the dataset, creating a filter that is a conjunction of conditions over all features.

4 Application to AH Flight Data

4.1 AH Classification Datasets

Using helicopters flight data, AH is developing tools such as virtual sensors, that aim at monitoring the aircraft health and usage. They use the historical flight data to learn a predictive model for a given parameter. The predicted value is compared to the one given by the physical sensor: an alert is raised if the difference between the two values is too high. An example of such a virtual sensor has been proposed by AH for the oil pressure of the helicopter Main Gear Box (MGB) [2]. We reuse the data from this study to perform the experiments of this paper. As a first contextualization had been done by AH domain experts, we used and compared two datasets, with 10 attributes selected and discretized by AH experts: the **raw dataset** corresponds to the flight data without any contextualisation, for a given period of time, randomly mixing tuples from several flights; the **expert-Contextualized dataset** is a subset of the raw one containing tuples filtered by AH experts (around 50% of the raw data).

4.2 Comparison of AH Datasets

The impact of contextualization was analyzed, by comparing accuracy for a random forest algorithm (*baseline* column of Table 1). The accuracy for the expert-contextualized dataset is much higher than for the raw one, confirming the expert contextualization pertinence. Moreover, $G_3 = 95.53\%$ for raw dataset and $G_3 = 95.51\%$ for expert-contextualized one. The proportion of counterexamples is therefore reasonable and the two datasets have similar G_3 values. The contextualization seems to have preserved the proportion of counterexamples: they have decreased in absolute number, but not with respect to the size of the dataset. New contextualization might therefore increase the model's accuracy.

Table 1. Accuracy of random forest models on the oil pressure datasets

Dataset	Baseline		Filter 1		Filter 2	
	# tuples	Accuracy	# tuples	Accuracy	# tuples	Accuracy
Raw	1969533	53.97%	607248	57.28%	468630	61.71%
Expert-contextualized	541342	73.94%	281947	76.02%	100165	78.61%

4.3 Additional Contextualization Using G_3

We applied our methodology from Fig. 1 to the two datasets, but first verified that the counterexamples were evenly distributed among the flights. Figure 3 shows a histogram of the percentage of counterexamples among flights: most flights have a very low rate of counterexamples, so any removal of counterexamples affects a large number of flights, avoiding the model to overfit on a subpart of the flights. We then analyzed the two plots made for each feature such as Fig. 4a and b for the pressure. Low pressure values have more counterexamples, while containing an important number of tuples. It can also be noted that the domain contextualization removes a significant part of counterexamples, but other regions could be cleaned further from counterexamples with additional contextualization, for example for pressure values over 5.6 bar.

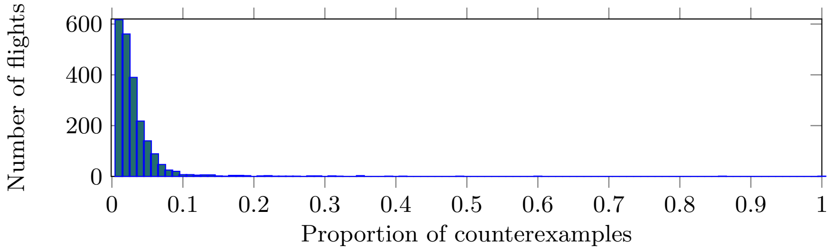


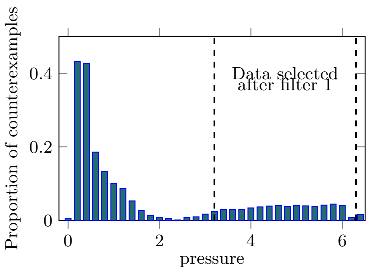
Fig. 3. Distribution of flights for each proportion of counterexamples

A first filter was designed (Filter 1 in Table 1). For the pressure, this filter removes all the data for which it is below 3.2 bar and above 6.4 bar, as these regions have few tuples but many counterexample (see Fig. 4). Similar rules were applied for the other features of the dataset. The results in Table 1 show the positive effect of this filter on classifier's accuracy. It was decided to improve again the contextualization, so we obtained filter 2 by adding additional rules to the ones from filter 1. Table 1 shows that accuracy is improved by filter 2. After this second iteration, the obtained contextualization was considered satisfying.

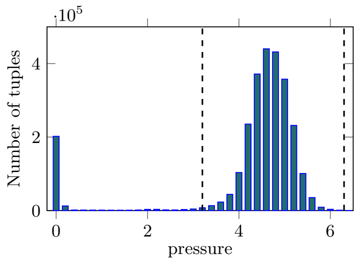
Finally, it should be noted from Table 1 that there is a significant gap between the highest accuracy on the raw dataset and the lowest accuracy for the expert-contextualized one. Even with the best filter, it is not possible to reach the result obtained using expert knowledge: the best approach consists in taking the valuable domain expert knowledge into account, before refining it using tools such as counterexamples and G_3.

4.4 Take Away Lessons

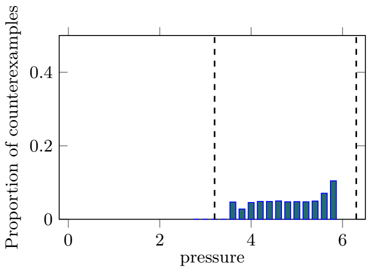
These experimentation showed how contextualization can be used to improve the accuracy of classifiers for AH virtual sensors. Contextualization is an important problem, but it is not easy to address because the proposed solutions are often domain-specific, or included in the “data preparation” steps that our left to data scientists judgment: our solution could in comparison be applied for other types of application and involves domain experts in the loop. There is also a qualitative aspect to this approach, that aims at taking a step back from the model, to understand what is being done, and understand the limitations. This is directly related to the explicability of the model, a crucial notion in aeronautics: the prediction of what can be seen as a simple classification algorithm output can put into question human lives getting back into an aircraft or not.



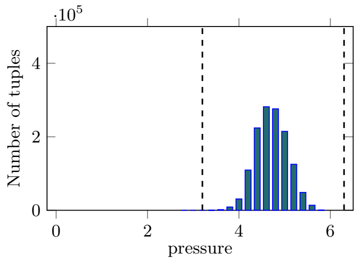
(a) Raw: counterexamples proportion against pressure



(b) Raw: histogram of pressure values



(c) Expert-contextualized: counterexamples proportion against pressure



(d) Expert-contextualized: histogram of pressure values

Fig. 4. Counterexamples and distribution for pressure values

5 Related Work

We applied our contextualization technique in the context of predictive maintenance for helicopters, a growing topic in the industry. Virtual sensors such as the ones used for the experiments of this paper [2] are interesting solutions in this context. Similarly, [3] proposes a virtual sensor to anticipate failures on photo-voltaic systems. Additionally, [11] presents a failure anticipation approach for aircraft systems. In this case, the learning is done only on flight phases predefined by experts. More generally, in most works developed in the industry, data is always combined with domain knowledge in order to speed-up accurate predictive models development. However, this combination still is often not optimal, and we believe this is a lever for improving accuracy of predictive models developed in the industry.

Functional dependencies are of high interest for data cleaning, a necessary prerequisite for data contextualization. The authors from [6] showed that if there is a functional dependency between features, it is likely to affect the classifier negatively. Specific dependencies have been proposed to identify inconsistencies in a dataset, and eventually repair it. Matching dependencies [4] for data repairing uses matching rules to relax the equality on functional dependencies and assign values for data repairing. In Holoclean [10], dependencies are used to clean automatically a dataset.

6 Conclusion

In this paper, we addressed the problem of contextualization of classification datasets, applied to the flight data of AH. This problem is crucial, and appears in many data science industrial applications, but has yet not been addressed as massively as other traditional machine learning problems. We proposed a methodology, and conducted experiments on data from a virtual sensor developed by AH, and showed how our method could improve the contextualization and, as a consequence, the accuracy of the datasets.

References

1. Arauzo-Azofra, A., Aznarte, J.L., Benítez, J.M.: Empirical study of feature selection methods based on individual feature evaluation for classification problems. Expert. Syst. Appl. **38**(7), 8170–8177 (2011)
2. Daouayry, N., Mechouche, A., Maisonneuve, P.L., Petit, J.M., Scuturici, M.: Data-centric helicopter failure anticipation: the MGB oil pressure virtual sensor case. In: International Conference on Big Data, p. 10 pages. IEEE (2019)
3. De Benedetti, M., Leonardi, F., Messina, F., Santoro, C., Vasilakos, A.: Anomaly detection and predictive maintenance for photovoltaic systems. Neurocomputing **310**, 59–68 (2018)
4. Fan, W.: Dependencies revisited for improving data quality. In: Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 159–170. ACM (2008)
5. Kivinen, J., Mannila, H.: Approximate inference of functional dependencies from relations. Theor. Comput. Sci. **149**(1), 129–149 (1995)
6. Kwon, O., Sim, J.M.: Effects of data set features on the performances of classification algorithms. Expert. Syst. Appl. **40**(5), 1847–1857 (2013). https://doi.org/10.1016/j.eswa.2012.09.017
7. Le Guilly, M., Petit, J.M., Scuturici, V.M.: Evaluating classification feasability over datasets using functional dependencies. In: BDA 2019 35ème conférence sur la Gestion de Données: Principes, Technologies et Applications. Lyon, France (2019)

8. Levene, M., Loizou, G.: A Guided Tour of Relational Databases and Beyond. Springer, New York (2012)
9. Mechouche, A., Daouayry, N., Camerini, V.: Helicopter big data processing and predictive analytics: Feedback and perspectives. In: Proceedings of the 45th European Rotorcraft Forum, Warsaw, Poland, p. 7 pages (2019)
10. Rekatsinas, T., Chu, X., Ilyas, I.F., Ré, C.: Holoclean: holistic data repairs with probabilistic inference. Proc. VLDB Endow. **10**(11), 1190–1201 (2017)
11. Sundareswara, R., Betz, F.D., Lu, T.C.: Interpretable unsupervised feature extraction and learning of abnormal system state transitions in aircraft sensor data. In: Proceedings of the Annual Conference of the PHM Society, vol. 10 (2018)

Query Intent Detection from the SEO Perspective

Samin Mohammadi(✉), Mathieu Chapon, and Arthur Frémond

Search ForeSight, 68 rue Marjolin, 92300 Levallois Perret, France
{samin.mohammadi,mathieu.chapon,arthur.fremond}@search-foresight.com

Abstract. Google users have different intents from their queries such as acquiring information, buying products, comparing or simulating services, looking for products and so on. Understanding the right intention of users helps to provide i) better content on web pages from the Search Engine Optimization (SEO) (Search engine optimization is the process of increasing the quality and quantity of website traffic by increasing the visibility of a website [1]) perspective and ii) more user-satisfying results from the search engine perspective. In this study, we aim to identify the user query's intent by taking advantage of Google results and machine learning methods. Our proposed approach is a clustering model that exploits some features to detect query's intent. A list of keywords extracted from the clustered queries is used to identify the intent of a new given query. Comparing the clustering results with the intents predicted by filtered keywords show the efficiency of the extracted keywords for detecting intents.

Keywords: Google search engine · Query intention · Search engine optimization

1 Introduction

Search engines try to predict users' intentions from their queries to provide the most accurate results. In the concept of search engines, intent detection generally is a classification problem aiming to find the intention of input data which is mainly in the format of text. Intent detection has different applications in Natural Language Processing (NLP) tasks such as question answering, chatbots, and search engines. In the tasks like question answering, the intent is normally one or several words selected from the question. While in the context of search engines, user queries are mainly divided into three groups in terms of their intent including *Informational*, *Navigational*, and *Transactional*. Informational queries are the most searched ones where the user's goal is looking for certain information by asking questions or searching for the keywords, for example "who is the CEO of Apple?". The user's intent from the navigational queries is to redirect to a specific website, for example "Apple". Transactional queries intend to do a transaction such as a purchase, for example "buy an iPhone".

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 49–59, 2020.
https://doi.org/10.1007/978-3-030-54623-6_5

Understanding the right intent of the user's query will help search engines to present the most related results which finally leads to higher user satisfaction. Plenty of researches have been done to identify query's intent [2,3]. Different types of models are proposed using machine learning [3,4] and natural language processing (NLP) models [5,6].

In this study, we look at the intent detection problem from the perspective of SEO which is different than a search engine's perspective. The main goal here is to identify the user's intent from what a search engine, such as Google, provides to users in response to their query. Identifying user's intent from this point of view will help the SEO to suggest better content for the websites. The suggested content will be aligned with what the search engine expects for a given query. It is finally speeding up and facilitating the semantic analysis from the SEO processing. The model's outputs are some set of keywords to filter the queries and label their intent easier and faster. The proposed model consists of five tasks 1) writing a scrapper and crawling the Google results from 2) extracting features 3) clustering queries against the extracted features, 4) characterizing clusters and find out their representing keywords, 5) comparing the results with manually annotated intents. The contributions of our model are as follows:

– Our crawler is able to scrape Google's results without getting blocked.
– It detects some intents which are different from the conventional intents.
– It can automatically detect the intent of a given query using keywords extracted from the clustering.

This paper is organized to present related works in Sect. 2, methodology and feature extraction in Sect. 3. We discuss the experiments in Sect. 4 and finally Sect. 5 concludes this study, its findings and discusses future works.

2 Related Works

Intent Detection: Intent detection generally is a classification problem aiming to find the intention of input data which is usually in the format of text. The models which are designed to identify intents base-on machine learning models can be divided into two groups [2] in terms of exploiting hand-crafted features [3,7] versus embedding features [5]. In the first group, researchers must extract some features which are important to identify query intent. Hand-crafted features can be derived from: i) Query tokens ii) Search Engine Results Pages (SERP) type and content tokens iii) interaction features (tracking user behavior including clicks log and queries log in a session).

In a study done by Sappelli [8], a dataset of user queries is collected with the features of the topic, action type, expected result type (image, video, map, etc.), location sensitivity and so on. They studied the distributions of queries per each feature as well as the correlations between different features. Authors in [9] took a transaction log from Dogpile into account to extract the required features such as query term, user id, time of day to train their clustering algorithm.

They finally characterized the identified clusters as informational, transactional and navigational and demonstrated the most frequent words for each category.

In a different study [7] Guo *et al.* investigated fine-grained user interactions with the search results to identify user intent. According to this study, although considering all features together provides 97% accuracy, features related to SERPs content are identified as the most important ones contributing to the classification. Transactional queries are identified by CURL in [10]. Authors in [11] trained a classifier to label queries with the intents extracted from user reviews. In [12] query-specific features, such as bag-of-words, length, recognized named entity, noun phrase, question and so on, are exploited to build three multi-class classifiers.

In the second group, the models are using embedding features automatically derived from mainly neural networks. For example, a convolutional neural network model is designed in [13] to extract the query embedding and use them to train the intent classifier. In an improved model of using word embedding, authors in [14] proposed a deep learning-based platform using Bi-directional Long-Short Term Memory (BDLSTM). They have used word embedding from GloVe [15] model and enriched them by bringing the synonyms and related words closer to each other in the vector space and moving the antonymous words away from each other. An automatic intent labeling model is introduced in [5] using Recurrent and Convolutional Neural Networks (RNN, CNN). The model is trained against ground truth and some heuristic rules to perform a multi-intent prediction for unlabeled queries. One of the latest models for intent detection is Zero-shot User Intent Detection via Capsule Neural Networks [16]. This model considers new and not-seen intentions also.

Two applications can benefit from intent detection researches, Search engines and SEO. Most of the mentioned researches that tried to detect the intent of a search query looking for solutions from the search engine perspective. Although SEOs can take advantage of those studies, dedicated research investigating the intent of queries from the SEO perspective is missing in the literature. Our proposed model uses hand-crafted features extracted from SERPs to cluster queries for the sake of SEO. Although some studies investigated SERPs' correlation to the query to identify the query's intent, no research has studied this problem from the SEO perspective.

Annotated Dataset: To build and train an intent detection model, a manually labeled dataset is needed. The model learns how to identify the intention of new data after getting trained by labeled data. According to [17], there are several difficulties in the intent detection task. The most important one is the lack of annotated datasets. Labeling the intent of queries is usually done manually. Besides the dataset, the intention of the user is not always explicit. The ambiguity and implicit intention make this problem more complicated and difficult.

In [5], almost 2k queries are manually labeled including both test and train datasets. Later, authors automatically labeled the rest of their data by a classifier trained on the ground truth labeled dataset. A big dataset of 30k queries randomly selected from AOL web queries is manually annotated in [12]. We use the last-mentioned dataset. We will discuss it in detail in Sect. 3.4.

3 Methodology

Early, we reviewed the previous studies on detecting the intent of queries. However, our research is different from past studies due to the following reasons:

1. It targets the features that have never been investigated in the literature.
2. Not only it will not rely on the manually annotated tags, (what most of the studies use them to design their classifiers) but also it will take advantage of a clustering model to verify the human-annotation of queries.
3. The Majority of the previous studies have addressed intent detection from the search engine's perspective. While this study looks at this problem from the perspective of SEO. Both are providing an automated method to identify the intent, but the second group provides additional advice to SEO to manage the content types which should be uploaded to the websites.

This subsection describes in detail how we take advantage of Google's search results to build a model for intent detection.

3.1 Data Scrapper

Data Scrapper is a python application, developed by our team, uses different techniques to send a query as a request to Google and Collects the results provided by Google. The script is written by Python and reads the queries from a public dataset provided in [12] to request them from Google and collect the provided results. The major challenge of Scrapper is not getting blocked by Google. When Scrapper requests so many queries from Google, it is faced by a captcha. The captcha should be filled by a human. Therefore, our solution is using a proxy to request queries from different IPs.

Scrapper collects different kinds of information from Google results shown in Table 1. This table shows each item with its description. To capture these features, Scrapper parses the HTML of the first page of the results. The results are saved in JSON files and then are processed to extract the final features. We will use that information in feature extraction and clustering processes.

3.2 Feature Extraction

Feature extraction is done after collecting Google results. We searched in the provided results for different types of results such as images, videos, featured snippets, rich snippets, knowledge graphs, direct answers, "people also ask" and so on. We consider each of those result types as a feature and extracted their title, number, and position. The idea behind this is to find what kind of information Google recognizes to show for each keyword. As the aim of SEO is to increase the visibility of a website and consequently a brand, thus, SEO consultants could easily decide what kind of content should be presented in the clients' website to get easily visible by Google. Understanding Google's methodology will lead to providing greater consultation for related and proper content on the website.

Table 1. Google's results types

Features	Description
Knowledge graph	It is a box that Google loads in the information related to each identified entity existing in the query from different sources
Calculator	It appears to answer directly the calculation-related queries
Direct answer	A box to respond a query that Google knows the answer
Map	Direct answer to map related questions
Local result	It shows the possibility of local access to the searched term
Commercial-sponsored	All the results showing the price
Twitter	If Google finds any tweets related to the searched term
Top stories	Google finds the recent news articles talking about the query
Videos	Very recent videos indicating the searched terms
Images	Categorized and recent images related to the query
Content navigation bar	Google provides a navigation bar of mainly objects such as movies, books on top of the search results
Featured snippet	It is a selected search result that answers the user's query right away. It can be a video, image, text, and so on
Rich snippets	Results in the form of cards having ratings and reviews
People also asked	A list of Questions similar to the searched query
Similar entity	A list of related entities to the searched entity
Google translator	Representing the meaning or translation
Top-button ads	Links with "Ad" next to their link
Natural results	Natural blue links on the first page (organic links)
Partners block	Links to Google's partners to search on their websites
Other cards	Boxes similar to the twitter block, such as popular products

As mentioned, features are extracted from different types of Google results. Our goal is to identify some groups of queries and consequently some keywords for which Google expects the websites to provide content in special formats.

3.3 Clustering

After feature extraction, now we need to apply a clustering model to find Informational, Transactional, and Navigational groups of keywords. As we pointed out before, there is a possibility that we end up in different categories than those three. To do clustering, we choose the KMeans algorithm [18]. The most proper number of clusters is found by the Elbow method to be 3.

3.4 Datasets

As we discussed in the state-of-the-art section, almost all the researches in this area use manually labeled data. Due to using clustering technique, our method does not need any labeled data. However, we use the labeled dataset to build an opportunity to compare the clustering results against the human-labeled tags. We use the public labeled dataset introduced in [12]. Table 2 shows the characteristics of the dataset used in our study. Majority of the queries in the selected-AOL dataset are informational.

Table 2. The dataset characteristics

Dataset name	#Queries	Manual-labels		
		Informational	Navigational	Transactional
Selected-AOL [12]	30k	23700	4574	1678

The Scrapper application crawls the public dataset's queries from Google and saves the results. It is worth mentioning that Google's SERPs are very user-dependent and its results are different from user to user. To have organic results, we run the scrapper on a server with neither search history nor logged in user. After crawling, we divided the dataset into the train and test sets with 90% and 10% population, respectively.

4 Clustering Experiments

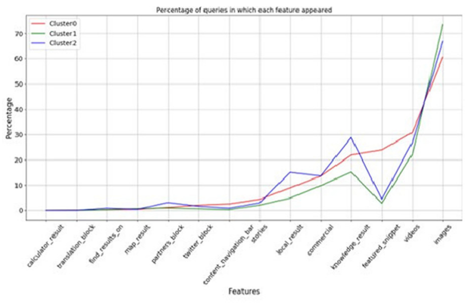
Before running clustering, we investigated the correlation of features to exclude the tightly correlated ones. Due to not founded any correlated features, we run the KMeans model with 19 features and K = 3. Characterizing clusters is ended up with very interesting clusters which are different from three predefined classes (including *Informational*, *Navigational*, and *Transactional*). Table 3 shows the distribution of queries and their tags in clusters. As an initial step to characterize the clusters, we study the value of the features for each cluster. We divided features into two groups, features with binary and numeric values. Figure 1a and b show the binary and numeric feature values for each cluster, respectively.

We first go through the binary features' plot. Each value in the plot refers to the percentage of the True values of each feature.

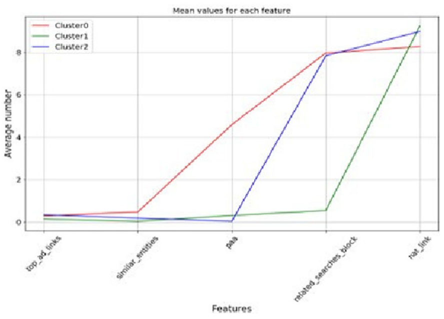
Table 3. Distribution of queries into clusters

Cluster name	#Queries	Informational	Transactional	Navigational
Cluster0	11582	9581	530	1471
Cluster1	5771	4392	472	907
Cluster2	9603	7339	529	1735

– Cluster0 has a noticeable higher value of featured snippets as well as a slightly higher value in the navigation bar feature. While the other two clusters have a very small value of featured snippet. It can be a piece of initial evidence for cluster0 of being information seeking queries.
– The only feature that has a higher value in cluster1 is *images*!
– Where two clusters 0 and 2 have almost the same values for commercial, this value for cluster1 is noticeably low. It shows low relation of queries in cluster1 with shopping intent.
– Cluster2 has a significantly high value of local results, knowledge results and somehow partners block, which is more likely to include queries having *Entities*[1] identified by Google and local information. Accompanying these two types of results can be interpreted as the queries that are looking for local special places. Later, looking at the words of queries will give more information about their exact intent.



Binary features value for each cluster



Numeric features value for each cluster

Fig. 1. Features values

In the numeric features' plot, the values indicate the mean value of features for each cluster. It also shows interesting points:

[1] The Google's Knowledge Graph has millions of entries that describe real-world entities like people, places, and things. These entities form the nodes of the graph, and are called Knowledge Graph Entities [19].

1. While the mean value of PAA is near to 0 for cluster1 and cluster2, cluster0's queries have an average value of 5. This observation strengthens the probability of cluster0 to be an informational cluster of queries.
2. In Fig. 1b, the related searches value for cluster1 is almost 0 which is strange! While for the other two clusters is almost 8 which is the regular number of suggestions by Google. To discover the reason, we have to look at the vocabulary of the queries in this cluster.

So far, from the observations of the features' values, we found cluster0 to include more informational (questions with a direct answer), and cluster2 more local queries. To find out more about the clusters, we plot the distribution of the words[2] of the queries in each cluster. In Fig. 2a, the bigger size of the vocabularies indicates more repetition. Big words such as {*new, best, americans*} shows that they get repeated more than other words in the queries of this cluster. Looking at the other less big words such as {*tax, business, car, education, health, house, college, university*} and putting them besides the most frequent ones lead to a representation of queries which mainly are searched to acquire information. Relying on the results so far discovers that Google tries to show featured snippet and PAA for this kind of general informational queries. In Fig. 2b, it is observable that the most frequent words are {*black, sex, women, nude, lyrics*}. Putting these words next to the other words in this group generates some sexual or racist phrases. The most highlighted feature of this cluster in our analysis from the previous section is almost zero number of keywords related to the query. The zero number of this feature for the queries of cluster1 illustrates that Google may not show similar keywords and queries to its users when their query carries a sexual or racist intent to may stop users from searching these kinds of queries.

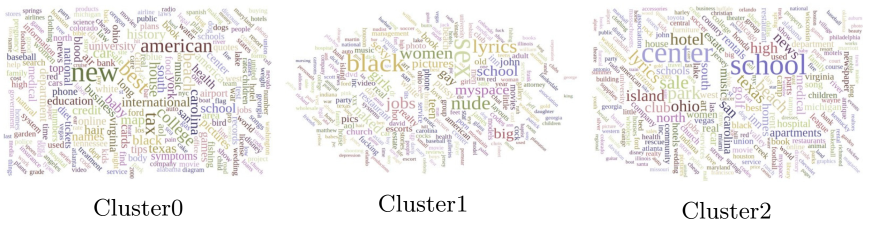


Fig. 2. Wordcloud distribution per cluster

Finally, in Fig. 2c, the most frequent vocabularies are {*center, school, park, beach, island, club, hotel, sale*}. Almost all of these words refer to locations. Based on our feature analysis, the dominant features of this cluster are *local result*, *knowledge graph* and *partners block*. The observations convey that cluster2 is mainly a collection of queries that are looking for local information including places (such as schools, hotels, islands, beaches, parks, and so on).

[2] https://www.jasondavies.com/wordcloud/.

Our model identified some intents (consisting of general qualitative information, racist/sexual intent, and local/place information) different than the conventional intents. The conventional intents are important for search engines. While the intents identified by our model is mainly for SEO. In a nutshell, the method is concluded to three clusters, i) Cluster0: general informative and qualitative queries, ii) Cluster1: queries with sexual and racist intent, iii) Cluster2: local and places information.

Although, all the queries grouped in different clusters are not exactly from the same context, the wordcloud representation of clusters can help us to extract some keywords to automatically tag the intent of a given query to be in one of the above clusters.

We processed the frequent words to exclude common, unrelated, and ambiguous words for each intent. To test the functionality of the extracted keywords, we automatically labeled the intent of the test queries based on their words. The test set with a 10% population is used for this experiment. In case of an equal number of words, we consider the maximum priority for Informational and the minimum priority for Sexual/Racism intents. From the other side, the queries inside the test dataset are scrapped from Google and the designed clustering model is applied on this dataset. The labels out of the clustering are compared to the labels out of the automatic labeling (vocabularies-filtering). Table 4 shows the results. Total number of queries is almost 3k with 2.4k *Informational*, 460 *Navigational*, and 150 *Transactional* intents.

Table 4. Clustering vs. vocabulary-based intent tagging results

		Vocabulary-based intents (predicted)				
		Informational	Local/Place information	Sexual/Racism	Precision	Recall
Clustering intents (Actual)	Informational	1232	54	25	0.46%	**0.94%**
	Local/Place information	904	141	26	**0.64%**	0.13%
	Sexual/Racism	519	25	70	**0.58%**	0.11%

In Table 4, the labels on the left side and the top are the outputs of clustering and vocabulary-based labeling, respectively. We consider the outputs of the clustering as the *actual labels* and compare them with the outputs of the vocabulary-based labeling as the *predicted labels*. Recall indicates the percentage of the queries in each intent which are predicted correctly inside that intent. According to the results, 94% of informational queries are correctly predicted as informational. While in the other intents, the low percentage of recall shows that most of their queries are assigned to some intents different than their intent. Two possible reasons can cause this low percentage, low accuracy of clustering and low efficiency of the extracted keywords. As we can not manually check the intent of each query, we will not be able to judge the accuracy of the clustering.

While investigating the second reason, we computed the precision values. High precision values for Local information and Sexual/Racism clusters indicate the efficiency of the keywords on identifying the right queries for those two intents. It means that the keywords are chosen precisely in such a way that they hardly misidentify the queries (with other intents) inside these two intents.

As a result, as we mentioned earlier, although some frequent keywords are representing each cluster, there remain some queries in each cluster that have none of their words in the representative keywords. Those queries are mislabeled in the Informational cluster using the vocabulary-based labeling. Consequently, the vocabulary-based method has a limitation which constraints it to label only the queries in which there are at least one of the selected keywords. This limitation persuades us to strengthen our model by taking other methods into account. As future work, we will take advantage of manual labels and Google's BERT model to identify the intents.

5 Conclusion

In this study, we have proposed and developed a model to identify the intent of user queries. The presented approach has the novelty of i) crawling Google SERPs results, ii) using new features (which have never been studied before), iii) identifying new and more detailed intents, iv) and finally, studying intent detection problem from the SEO perspective. Although Google has previously provided very few words for each identified intents, without any solid method to extend those words, our clustering model can provide three sets of keywords to automatically identify the query's intent.

As a future work, we will study the semantic and NLP relation between the query and those features that have text, such as featured snippets, PAA, and knowledge graph. We will use as well the fine-tuned BERT model to identify the intention of user queries and compare the results with the baseline methods.

References

1. Search engine optimization. https://en.wikipedia.org/wiki/Search_engine_optimization
2. Zhou, S., Cheng, K., Men, L.: The survey of large-scale query classification. In: AIP Conference Proceedings, vol. 1834, no. 1, p. 040045. AIP Publishing (2017)
3. Jansen, B.J., Booth, D.L., Spink, A.: Determining the user intent of web search engine queries. In: Proceedings of the 16th International Conference on World Wide Web, pp. 1149–1150. ACM (2007)
4. Ashkan, A., Clarke, C.L.A., Agichtein, E., Guo, Q.: Classifying and characterizing query intent. In: Boughanem, M., Berrut, C., Mothe, J., Soule-Dupuy, C. (eds.) ECIR 2009. LNCS, vol. 5478, pp. 578–586. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00958-7_53
5. Pîrvu, M.C., Anghel, A., Borodescu, C., Constantin, A.: Predicting user intent from search queries using both CNNs and RNNs. arXiv:1812.07324 (2018)

6. Meng, L., Huang, M.: Dialogue intent classification with long short-term memory networks. In: Huang, X., Jiang, J., Zhao, D., Feng, Y., Hong, Y. (eds.) NLPCC 2017. LNCS (LNAI), vol. 10619, pp. 42–50. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73618-1_4
7. Guo, Q., Agichtein, E.: Ready to buy or just browsing?: detecting web searcher goals from interaction data. In: Proceedings of the 33rd International ACM SIGIR, pp. 130–137. ACM (2010)
8. Sappelli, M., Verberne, S., Heijden, M.v.d., Hinne, M., Kraaij, W.: Collection and analysis of ground truth data for query intent (2012)
9. Kathuria, A., Jansen, B.J., Hafernik, C., Spink, A.: Classifying the user intent of web queries using k-means clustering. Internet Research (2010)
10. Sun, Y., Loparo, K.: A clicked-URL feature for transactional query identification. In: 43rd Annual Computer Software and Applications Conference. IEEE (2019)
11. Boteanu, A., Dutile, E., Kiezun, A., Artzi, S.: Subjective search intent predictions using customer reviews (2020)
12. Figueroa, A.: Exploring effective features for recognizing the user intent behind web queries. Comput. Ind. **68**, 162–169 (2015)
13. Hashemi, H.B., Asiaee, A., Kraft, R.: Query intent detection using convolutional neural networks. In: International Conference on Web Search and Data Mining, Workshop on Query Understanding (2016)
14. Sreelakshmi, K., Rafeeque, P., Sreetha, S., Gayathri, E.: Deep bi-directional lstm network for query intent detection. Procedia Comput. Sci. **143**, 939–946 (2018)
15. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
16. Xia, C., Zhang, C., Yan, X., Chang, Y., Yu, P.S.: Zero-shot user intent detection via capsule neural networks. arXiv preprint arXiv:1809.00385 (2018)
17. Liu, J., Li, Y., Lin, M.: Review of intent detection methods in the human-machine dialogue system. J. Phys. Conf. Ser. **1267**(1), 012059 (2019)
18. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, vol. 1, no. 14, pp. 281–297 (1967)
19. Define and refine search using knowledge graph entities. https://support.google.com/customsearch/answer/9350760?hl=en

Fast and Accurate Group Outlier Detection for Trajectory Data

Youcef Djenouri[1(✉)], Kjetil Nørvåg[2], Heri Ramampiaro[2], and Jerry Chun-Wei Li[3]

[1] Department of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway
youcef.djenouri@sintef.no

[2] Department of Computer Science, NTNU, Trondheim, Norway
{noervaag,heri}@ntnu.no

[3] Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway
jerrylin@ieee.org

Abstract. Previous approaches to solve the trajectory outlier detection problem exclusively examine single outliers. However, anomalies in trajectory data may often occur in groups. This paper introduces a new problem, *group trajectory outlier detection* (GTOD) and proposes a novel algorithm, named, CDkNN-GTOD (**C**losed **D**BSCAN **kN**earest **N**eighbors for **G**roup **T**rajectory **O**utlier **D**etection). The process starts by determining micro clusters using the DBSCAN algorithm. Next, a pruning strategy using kNN is performed for each micro cluster. Finally, an efficient pattern mining algorithm is applied to the resulting subsets of group of trajectory candidates to determine the group of trajectory outliers. We performed a comparative study using real trajectory databases to evaluate the proposed approach. The results have shown the efficiency and effectiveness of CDkNN-GTOD.

Keywords: Group Trajectory Outlier Detection · Pattern mining · Clustering

1 Introduction

The proliferation of GPS devices has resulted in countless of sequence points representing trajectories being generated, stored, and analyzed in the context of urban data [4]. Without loss of generality, in the context of intelligent transportation, the data analyst is faced with a myriad of trajectories derived from the mobility of people, cars, buses, taxis, among others. Previous approaches to solve the trajectory outlier detection have solely considered *individual* outliers. In real-world applications, however, trajectory outliers often appear in groups, e.g., a group of bikes that deviates to the usual trajectory due to the maintenance of streets. This paper presents a new problem of trajectory outlier detection called *Group Trajectory Outlier Detection* (GTOD), which the goal is to identify group of anomalous behaviours from trajectory data.

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 60–70, 2020.
https://doi.org/10.1007/978-3-030-54623-6_6

Motivation and Idea. Consider the example of taxi trajectories, each trajectory is mapped to the road map network. Traditional trajectory outlier detection algorithms, e.g., [1], may detect individual outliers. However, these algorithms cannot identify outliers, where a group of taxis deviate from the usual trajectory. Detecting such trajectory outliers, could help (taxi) planners to study the different correlations between these trajectories to deduce useful information. For example, a group of taxi trajectory outliers could indicate that the taxis are partners in a taxi fraud. However, by observing only *individual* deviations, such a possible fraud would be hard to reveal. Motivated by the limitation of solely identifying individual outliers, we focus on studying, and determining group of trajectory outliers. To do so, we first define a new problem called *Group Trajectory Outlier Detection*, and then propose a novel approach for finding these kind of anomalies. The process starts by determining the micro clusters, using DBSCAN, each micro cluster is considered as a candidate group of trajectory outliers. Each group contains several individual trajectory outliers that are close to each other. Note that the groups may contain normal trajectories as well. Such trajectories can generally be considered as noises. To remove such noises, the set of group of trajectory outliers are pruned using the kNN algorithm. Finally, we run a pattern mining algorithm to explore the correlation among the pruned groups of trajectory outliers. The discovered frequent patterns are thereafter considered as the final groups of trajectory outliers.

Contribution. This paper presents a new problem called *Group Trajectory Outlier Detection* (GTOD for short), which allows to identify groups of trajectory outliers. The main contributions of the presented work can be summarized as follows. i) We introduce and formulate a new problem called *GTOD: Group Trajectory Outlier Detection* to enable to identify group of trajectory outliers. ii) We propose a new technique, named CDkNN-GTOD (**C**losed **D**BSCAN **kN**earest **N**eighbors for **G**roup **T**rajectory **O**utlier **D**etection), which explores the DBSCAN algorithm for determining candidate outliers represented as micro clusters, kNN algorithm for pruning the micro clusters, and a pattern mining process for discovering the group of trajectory outliers. iii) We demonstrate the performance of the proposed algorithm using different real trajectory databases. The results of experiments reveal that CDkNN-GTOD outperforms the baseline algorithms for group outlier detection.

2 Related Work

Chalapathy et al. [2] proposed the deep generative model to find out the group outliers on various image applications. The outlierness for each group in the input data was then estimated by group reference function using the backpropagation algorithm. Liang et al. [12] developed a flexible genre model to find specific group outliers. Their main idea was to characterize data groups at both point and group level to detect various types of anomalous groups. Das et al. [3] explored the different correlations between data outliers to detect anomalous patterns using Bayesian network anomaly detection and conditional anomaly detection. Xiong et al. [11] proposed a group outlier detection approach by defining a mixture of Gaussian mixture model. It adopted the likelihood of each group, the marginal likelihood of each observation within a group, and the maximum likelihood estimation to learn the hyperparameters of the mixture model. Soleimani et al. [9] developed a supervised learning approach that groups anomalous patterns when memberships are previously unknown. The salient features were extracted from an appropriate training set with discrete data inputs. Li et al. [7] assigned feature weights on each group outlier, and computed chain rule entropy to determine correlation between different feature groups. Toth et al. [10] reviewed both static, and dynamic group anomaly detection solutions. The static group anomaly detection is the process of identifying groups that are not consistent with regular group patterns, while dynamic group change detection assesses significant differences in the state of a group over a period of time. In contrast to this, in this study, we are interested in dealing with static group anomaly detection on the trajectory data. From this brief review, we can conclude that approaches to group outlier detection algorithms are mainly based on some known distributions to find group outliers. In real scenarios, it is hard to fit the data to such distributions. In this paper, we introduce a new problem called group of trajectory outlier detection and propose a new data mining approach, which do not need to know the distribution of the input data to determine the group of trajectory outliers.

3 Problem Statement

Definition 1 (Trajectory Database). *We define a trajectory database $T = \{T_1, T_2 ... T_m\}$, where each raw trajectory T_i is a sequence of spatial location points $(p_{i1}, p_{i2} ... p_{in})$, obtained by localization techniques such as GPS. Each point is represented by the latitude, and the longitude values, respectively.*

Definition 2 (Mapped Trajectory Database). *We define a mapped trajectory database $\Lambda = \{\Lambda_1, \Lambda_2 ... \Lambda_m\}$, where each mapped trajectory Λ_i is a sequence of spatial location regions $(R_{i1}, R_{i2} ... R_{in})$, obtained by mapping each point in T_i to the closest region R_i. We note $R = \{R_1, R_2 ... R_{|R|}\}$, by the set of all regions.*

Definition 3 (Trajectory Dissimilarity). *We define the distance between two trajectories $d(\Lambda_i, \Lambda_j)$ by the number of all regions minus the number of shared regions between the two trajectories Λ_i, and Λ_j, as*

$$d(\Lambda_i, \Lambda_j) = n - |\{(R_{il}, R_{jl}) | R_{il} = R_{jl}, \forall l \in [1..n]\}| \tag{1}$$

Definition 4 (Group Trajectory Candidate). *We define a group of trajectory candidate $\mathcal{G}$ by the set of individual trajectory outliers retrieved from the set of individual trajectory outliers ITO, i.e.,*

$$\mathcal{G} = \{\Lambda_i | \Lambda_i \in ITO\} \tag{2}$$

Definition 5 (Density Group). *We define the density of the candidate group trajectory outliers $\mathcal{G}$ as*

$$Density(\mathcal{G}) = \frac{|\mathcal{G}|}{|\{R_j | \Lambda_i \in \mathcal{G}, R_j \in \Lambda_i\}|} \tag{3}$$

To normalize the density function, we divide the result by the density of the group having maximum density value, this ensures to obtain values ranged from 0 to 1. We call this function NormalizedDensity.

Definition 6 (Group Trajectory Outlier). *A set of trajectories $\mathcal{G}$ is called a Group Trajectory Outlier if and only if,*

$$\begin{cases} \mathcal{G} \subseteq ITO \\ NormalizedDensity(\mathcal{G}) \geq \gamma \end{cases} \tag{4}$$

Note that γ is the density threshold varied from $[0 \ldots 1]$.

Definition 7 (Non-Redundant Group Trajectory Outlier). *A group of trajectory outliers $\mathcal{G}$ is called a Non-Redundant Group Trajectory Outlier if it has no superset of $\mathcal{G}$, that is a group of trajectory outlier.*

Definition 8 (Group Trajectory Outlier Detection Problem). *Group Trajectory Outlier Detection Problem aims to discover from the set of all mapped trajectories, the set of all non-redundant groups of trajectory outliers, denoted by $\mathcal{G}^*$.*

4 CD*k*NN-GTOD Algorithm

This section presents our algorithm CD*k*NN-GTOD, (Closed DBSCAN k Nearest Neighbors for Group Trajectory Outlier Detection). Our main goal is to efficiently explore the enumeration tree of the trajectory candidates to determine the group of trajectory outliers. In this work, we inspire by the clustering, the neighborhood computation, and the pattern mining algorithms to accurately prune the search space and find the group of trajectory outliers. The process starts by finding the micro clusters using DBSCAN algorithm, the pruning strategy is performed for each micro cluster using the *k*NN principle. An efficient pattern mining algorithm is then explored on the resulted subset of group of trajectory candidates to determine the groups of trajectory outliers. In the remaining of this section, we show how to use all these concepts in the CD*k*NN-GTOD framework.

4.1 Clustering

Before presenting the clustering step, we need formally define some basic concepts.

Definition 9 (Trajectory Neighborhoods). *We define the neighborhoods of a trajectory Λ_i, $\mathcal{N}_{\Lambda_i}$, for a given threshold ϵ by*

$$\mathcal{N}_{\Lambda_i} = \{\Lambda_j | d(\Lambda_i \Lambda_j) \leq \epsilon \vee j \neq i\} \tag{5}$$

Definition 10 (Core Trajectory). *A trajectory Λ_i is called core trajectory if there is at least a minimum number of trajectories MinPts such that $|\mathcal{N}_{\Lambda_i}| \geq$ MinPts*

Definition 11 (Micro Cluster). *A cluster of trajectories C_i is called a micro cluster if and only if $0 < |C_i| \leq \mu$, where μ is a user threshold.*

This section presents how to use *DBSCAN* algorithm to identify micro clusters, each micro cluster is considered as group of trajectory outlier candidates. The ϵ-neighborhood of each trajectory is computed using Definition 9. The core trajectories are determined using Definition 10. DBSCAN then iteratively collects density-reachable trajectories from these core trajectories directly, which may involve merging a few density-reachable clusters. The process terminates when no new trajectories can be added to any cluster. Initially, the set of trajectories are grouped using *DBSCAN*. This generates several clusters with different sizes. Each micro cluster (see Definition 11) is considered as group candidates. As a result, sets of groups trajectory candidates called $\{\mathcal{G}_i^+\}$ are generated.

4.2 Pruning Strategy

The clustering step returns micro clusters, where each micro cluster forms the groups of trajectory candidates. These groups contain individual trajectory outliers close to each other. However, they may contain normal trajectories. To well prune the groups trajectory candidates, we develop an efficient pruning strategy based on kNN principle. Before presenting the pruning step, we need formally define some basic concepts.

Definition 12 (kNN of a trajectory). *We define kNN of a trajectory Λ_i, denoted by kNN(Λ_i) as*

$$kNN(\Lambda_i) = \{\Lambda_j \in \Lambda \setminus \{\Lambda_i\} | d(\Lambda_i, \Lambda_j) \leq k_{dist}(\Lambda_i)\} \tag{6}$$

$k_{dist}(\Lambda_i) = d(\Lambda_i, \Lambda_l)$ *is the k-distance of the trajectory Λ_i defined such as it exists k trajectories $\Lambda' \in \Lambda$, it holds that $d(\Lambda_i, \Lambda_l) \geq d(\Lambda_i, \Lambda')$*

Definition 13 (Outlierness degree of a Trajectory). *We define the outlierness degree of a given trajectory Λ_i, denoted by $\delta(\Lambda_i)$ as*

$$\delta(\Lambda_i) = |\{\Lambda_j | j \neq i \vee \Lambda_j \in (kNN(\Lambda_i) \cap \mathcal{G}^+)\}| \tag{7}$$

In the following, we present an adapted kNN algorithm for pruning the candidate trajectory outliers. The algorithm considers as input the sets of all trajectory candidate $\mathcal{G}^+$. The process aims to reduce the number of candidate trajectory outliers on each micro cluster. For each micro cluster, it first adds the trajectory outlier with highest outlierness degree, Λ_1^+, to the set of candidate trajectory outliers labeled by Λ_1^+, and denoted by $\mathcal{G}_1^+$. It then generates all potential candidates from Λ_1^+. A trajectory t is a potential candidate from Λ_1^+, if and only if, $t \in \mathcal{G}_1^+ \vee t \in k\text{NN}(\Lambda_1^+)$. The same process is recursively applied for all potential candidates added to $\mathcal{G}_1^+$, and the overall process is repeated for all micro clusters.

4.3 Pattern Mining

Consider GTOD problem $<R, \mathcal{G}^+, \mathcal{G}^*, NormalizedDensity(\bullet), \gamma>$, it could be fit to the pattern mining problem [6] represented by the set of all transactions D, the set of items I, the support function $Support$, the minimum support $minsup$, and the set of all returned patterns P, as follows,

$$D = R, I = \mathcal{G}^+, Support(\bullet) = NormalizedDensity(\bullet), minsup = \gamma, \mathcal{G}^* = P$$

Each region is viewed as a transaction, and each trajectory candidate is viewed as an item. A pattern is a subset from $\mathcal{G}^+$ already pruned. The support of the pattern p is equal to the density of the group of trajectories of p. The minimum threshold will be γ threshold. A pattern mining process is applied on the set of transactions D, and the set of items I, with the support function $NormalizedDensity(\bullet)$, and with the minimum support set to γ. Each frequent pattern discovered is considered as a set of group of trajectory outliers. By definition, GTOD problem aims to identify **non-redundant** group of trajectory outliers. If we apply classical pattern mining algorithm [5], redundant patterns may be extracted. To deal with this issue, we aim to discover closed patterns, this ensures non-redundant group of trajectory outliers are derived. In our implementation, we used Closet algorithm [8] to find out the closed patterns. It proceeds in two steps. Initially, all closed frequent patterns of size 1 are mined. Then, new patterns are generated by directly working on the closed frequent patterns of size 1, without mining additional frequent patterns. It used sparse two efficient data structures id-lists and vertical id-lists for fast counting the support of closed frequent patterns and one-step technique to prune the search space and check the closure property.

5 Performance Evaluation

Extensive experiments have been carried out to compare the CDkNN-GTOD algorithm with the state-of-the art group outlier detection algorithms. The evaluation is performed using ROCAUC, which is common measure for the evaluation of outlier detection methods. We perform the experiments using well-known trajectory databases, retrieved from different repositories, consisting of the following: Geolife[1], Manhattan[2], ECML PKDD 2015 competition[3], and big taxi trajectories: taxi 13-1, taxi 13-2, and taxi 15 [13].

5.1 Parameter Settings

The first part of this experiment focuses on tuning the parameters of different stages of CDkNN-GTOD algorithm. It is performed on two parts, the first one

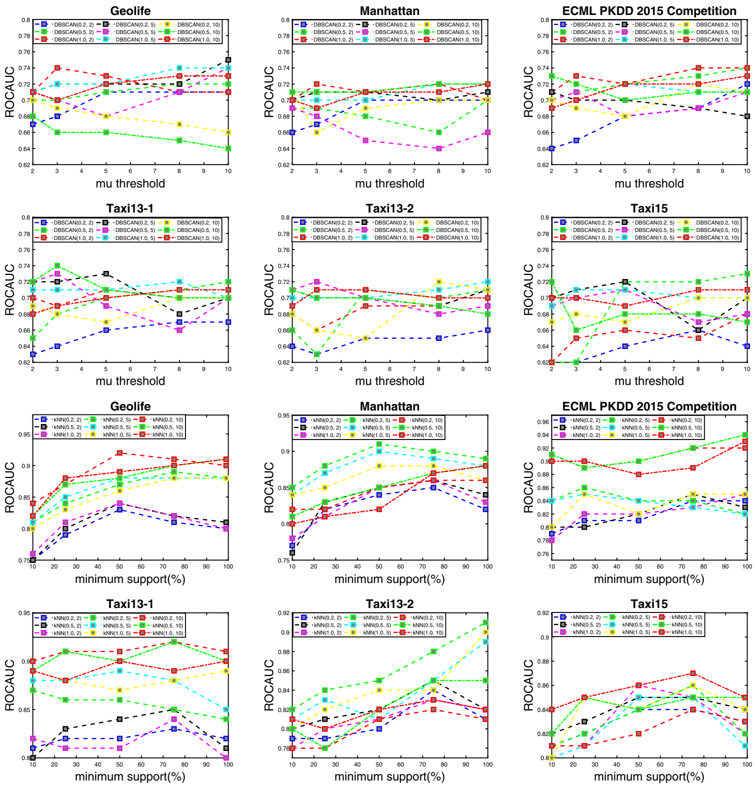


Fig. 1. The parameter setting of the CDkNN-GTOD

is to tune the parameters of the clustering step represented by the DBSCAN parameters (ϵ, and MinPts), μ for determining the micro clusters, the second one is to tune the parameters of kNN represented by the number of neighborhood,

[1] https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-data-set-user-guide/.
[2] https://lab-work.github.io/data/.
[3] http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html.

k, and the density threshold γ, and the parameter of the pattern mining step represented by the minimum support threshold, minsup. Figure 1 shows the first part of the parameters setting, by considering the micro clusters retrieved in the clustering step as group of trajectory outliers, and ignoring the pruning and the pattern mining processes. Several tests have been performed using different trajectory databases by varying the DBSCAN parameters, ϵ from 0.2 to 1.0, and MinPts from 2 to 10, the μ parameter for determining the micro clusters from 2 to 10. Whatever the trajectory database used as input, the accuracy determined by the ROCAUC value exceeds 0.72, however does not go up 0.75. These results are explained by the fact that the idea of the micro clusters is able to identify the group of trajectory outliers but not in an optimal way. Therefore, in the next experimentation, we tune the parameters of the pruning and the pattern mining processes, by fixing the best parameters of the clustering step for each trajectory database found in this part. The results of the second part is highlighted in Fig. 1, we varied the number of neighborhood from 2 to 10, the density threshold values from 0.2 to 1.0, and the minimum support values from 10% to 99%. The results reveal that the pruning and the pattern mining steps improve the accuracy of the proposed algorithm. This is explained by the fact that kNN strategy allows to prune the search and keep only the most neighbors of trajectory outliers in the micro clusters. Moreover, the pattern mining process further reduces the search space by exploring the frequent patterns among the group of trajectory outliers in the micro clusters. Table 1 summarizes the best parameters values of the CDkNN-GTOD algorithm, which will be used in the remaining of the experiments.

Table 1. Best parameters of CDkNN-GTOD.

Database	ϵ	MinPts	μ	k	γ	Minsup
Geolife	0.2	5	10	10	0.2	50
Manhattan	0.5	10	8	5	0.2	50
ECML PKDD 2015 competition	1.0	10	8	10	0.5	99
Taxi13-1	0.5	10	3	10	0.5	75
Taxi13-2	0.5	5	3	10	0.5	99
Taxi15	0.5	10	10	10	1.0	75

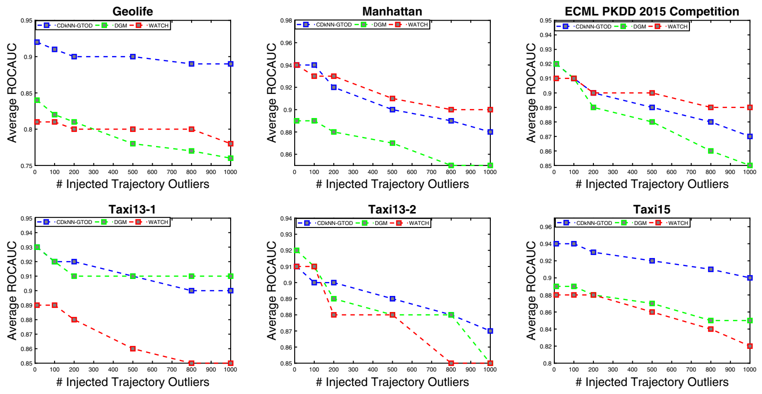


Fig. 2. CDkNN-GTOD vs. state-of-the-art group outlier detection algorithms: accuracy

5.2 CDkNN-GTOD Vs State-of-the-Art Group Detection Algorithms

The aim of this experiment is to compare CDkNN-GTOD with the baseline algorithms in terms of accuracy and processing time. To the best of our knowledge, this is the first work which investigates the group outlier detection in trajectory data. Therefore, we adopt two baseline group outlier detection algorithms (DGM [2], and WATCH [7]) to trajectory data for comparison with CDkNN-GTOD. Figure 2 presents the average ROCAUC value of the proposed algorithm CDkNN-GTOD, and the baseline group outlier detection algorithms (DGM and WATCH), using several trajectory databases, and with different number of injected outliers. By varying the number of injected trajectories from 10 to 1000, the CDkNN-GTOD outperforms the other algorithms for almost of cases. Among 36 cases shown, CDkNN-GTOD is the best for 22 cases, DGM for 8 cases, and WATCH for 6 cases. Moreover, when increasing the number of injected trajectory outliers, the accuracy of the CDkNN-GTOD stabilizes and do not go under 0.87, whereas, the accuracy of the baseline algorithm goes under 0.80. This comes from the fact that our approach uses more advanced and recent strategies, based on clustering, neighborhoods, and pattern mining, while the baseline approaches use less advanced concepts of outlier detection based on data distribution. Regarding processing speed, as shown in Fig. 3, our approach is very competitive compared to the baseline approaches. This is explained the way we combined the efficient data mining techniques – clustering, kNN, and pattern mining, for finding the groups of trajectory candidates.

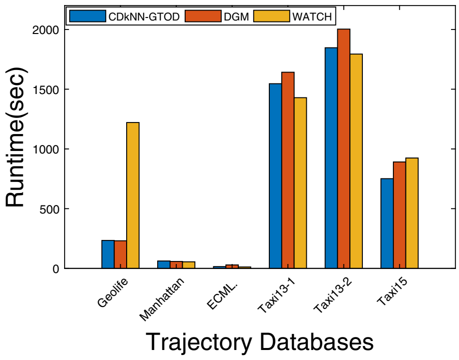


Fig. 3. CDkNN-GTOD vs. state-of-the-art group outlier detection algorithms: runtime

6 Conclusion

In this paper, we introduced a new problem that aims at discovering group of trajectory outliers. To solve this problem we proposed to combine clustering, pruning, and pattern mining. More specifically, our approach consisted of three main steps: (1) determination of micro clusters using the DBSCAN algorithm, (2) identification of potential group of trajectory candidates from the micro clusters with kNN, and (3) pruning of the candidates using density computation/pattern mining. Each of these steps are executed in an iterative manner, allowing to extract the group of trajectory outliers in an effective and efficient manner. To evaluate our approach, we performed our comparative experiments on different real trajectory databases. The experiments showed that our approach achieved good results in terms of both accuracy and processing speed. Overall, the proposed approach is indeed capable of effectively and efficiently solving the GTOD problem, and that it outperforms traditional methods which are based on data distribution. Nevertheless, the combination of the advanced techniques requires high expertise not only in trajectory analysis or outlier detection, but in other sophisticated data mining techniques. In our future work, we will investigate and target new applications of *GTOD*, such as climate change analysis, e.g., finding a group of hurricane trajectories that deviates from the normal hurricane ones. This would allow to early identify other cities that could be affected.

References

1. Belhadi, A., Djenouri, Y., Lin, J.C.W.: Comparative study on trajectory outlier detection algorithms. In: 2019 International Conference on Data Mining Workshops (ICDMW), pp. 415–423. IEEE (2019)
2. Chalapathy, R., Toth, E., Chawla, S.: Group anomaly detection using deep generative models. In: Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., Ifrim, G. (eds.) ECML PKDD 2018. LNCS (LNAI), vol. 11051, pp. 173–189. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10925-7_11

3. Das, K., Schneider, J., Neill, D.B.: Anomaly pattern detection in categorical datasets. In: Proceedings of the 14th ACM SIGKDD, pp. 169–176 (2008)
4. Djenouri, Y., Belhadi, A., Lin, J.C.W., Djenouri, D., Cano, A.: A survey on urban traffic anomalies detection algorithms. IEEE Access **7**, 12192–12205 (2019)
5. Djenouri, Y., Djenouri, D., Lin, J.C.W., Belhadi, A.: Frequent itemset mining in big data with effective single scan algorithms. IEEE Access **6**, 68013–68026 (2018)
6. Djenouri, Y., Lin, J.C.W., Nørvåg, K., Ramampiaro, H.: Highly efficient pattern mining based on transaction decomposition. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 1646–1649. IEEE (2019)
7. Li, J., Zhang, J., Pang, N., Qin, X.: Weighted outlier detection of high-dimensional categorical data using feature grouping. IEEE Trans. Syst. Man Cybern. Syst. **99**, 1–14 (2018)
8. Pei, J., Han, J., Mao, R., et al.: CLOSET: an efficient algorithm for mining frequent closed itemsets. In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, vol. 4, pp. 21–30 (2000)
9. Soleimani, H., Miller, D.J.: ATD: anomalous topic discovery in high dimensional discrete data. IEEE Trans. Knowl. Data Eng. **28**(9), 2267–2280 (2016)
10. Toth, E., Chawla, S.: Group deviation detection methods: a survey. ACM Comput. Surv. (CSUR) **51**(4), 77 (2018)
11. Xiong, L., Póczos, B., Schneider, J., Connolly, A., VanderPlas, J.: Hierarchical probabilistic models for group anomaly detection. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 789–797 (2011)
12. Xiong, L., Póczos, B., Schneider, J.G.: Group anomaly detection using flexible genre models. In: Advances in Neural Information Processing Systems, pp. 1071–1079 (2011)
13. Zhang, D., Li, N., Zhou, Z.H., Chen, C., Sun, L., Li, S.: iBAT: detecting anomalous taxi trajectories from GPS traces. In: Proceedings of the 13th International Conference on Ubiquitous Computing, pp. 99–108 (2011)

Data Processing

On the Performance Impact of Using JSON, Beyond Impedance Mismatch

Moditha Hewasinghage(✉), Sergi Nadal, and Alberto Abelló

Universitat Politècnica de Catalunya (BarcelonaTech), Barcelona, Spain
{moditha,snadal,aabello}@essi.upc.edu

Abstract. NOSQL database management systems adopt semi-structured data models, such as JSON, to easily accommodate schema evolution and overcome the overhead generated from transforming internal structures to tabular data (i.e., impedance mismatch). There exist multiple, and equivalent, ways to physically represent semi-structured data, but there is a lack of evidence about the potential impact on space and query performance. In this paper, we embark on the task of quantifying that, precisely for document stores. We empirically compare multiple ways of representing semi-structured data, which allows us to derive a set of guidelines for efficient physical database design considering both JSON and relational options in the same palette.

1 Introduction

The relational model was defined as an abstraction level to gain independence of the file system and any internal storage structure [6]. Thus, we could gain flexibility and interoperability without losing efficiency by following a tabular representation and some normal forms. Indeed, the first normal form (1NF) established that attribute domains had to be atomic (i.e., they could be neither compound-complex structures nor arrays). However, a rigid tabular structure is not adequate in modern agile software development, where the schema is under continuous evolution. Moreover, a well-known problem of RDBMS is the impedance mismatch, defined as the overhead generated by transformations from internal structures to tables, and then into programming structures [3].

The development of NOSQL systems, which adopt more flexible data representations, allowed to overcome the impedance mismatch [14]. Such data formats (e.g., JSON), are directly mapped from disk to memory. This is additionally achieved by breaking 1NF, allowing typical programming nested structures and arrays in the attribute values (e.g., MongoDB encourages denormalization[1]). Furthermore, such semi-structured formats, also allow to skip schema declaration, which is beneficial in highly evolving applications [13]. Nevertheless, it is not clear whether denormalization and schemaless is a conscious design choice,

[1] https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-2.

Partly funded by the European Commission through the programme "EM IT4BI-DC". We thank Braulio Blanco for assisting on the first version of the experiments.

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 73–83, 2020.
https://doi.org/10.1007/978-3-030-54623-6_7

or merely a paradigm imposed by the limitations of NOSQL systems. Yet, the flexibility offered by NOSQL comes at a price, where each one of the associated design choices may widely change their physical representation, and thus profoundly impact performance. Practitioners have ignored this, and today make binary design decisions based on rules, and programming needs with no overall view of the system needs [12]. Thus, it is vital to consider the benefits and drawbacks posed by these different alternatives during the design process [5]. Relational and semi-structured data models, are not a simple binary choice, but a continuum of options with different degrees of (de)normalization.

In this paper, we quantify the performance impact of physical database design choices on NOSQL systems, focusing on the JSON data model. To this end, different design choices (i.e., equivalent representational differences) related to both metadata (i.e., schema), such as attribute embedding or optionality, and data, such as nested objects or arrays, are quantitatively scrutinized. We acknowledge that many DBMS features can affect performance (i.e., concurrency control and recoverability mechanism, distribution and parallelism management, connection pools and setup, etc.). Nevertheless, we only study the impact of design decisions on a semi-structured data model, being agnostic of the technological choice. Our main contributions are as follows: (1) We identify the main physical design characteristics of semi-structured data and compare them to their structured counterpart. (2) We empirically quantify the impact of design choices in semi-structured data. (3) We evaluate the different designs in a relational and NOSQL DBMS.

The rest of the paper is structured as follows. Section 2 discusses related work. Section 3 presents design differences. Section 4 shows experimental results. Sections 5 and 6 discuss the experimental findings and conclude the paper.

2 Related Work

[4] abstracts and homogenizes the modeling commonalities of NOSQL systems. It considers databases as sets of collections, which in turn are sets of blocks, finally represented by sets of entries. Similarly, [9] proposes a subject-oriented methodology to design NOSQL databases. A conceptual model of the system is converted into an equivalent hypergraph representation, such that hyperedges identify specializations or aggregations among entities. For each hyperedge, an specific data model, either relational or co-relational. [7] proposes a method to generate NOSQL databases from a high-level conceptual model automatically. The authors propose the UML-like Generic Data Metamodel, integrating structural and data access patterns. Then, a set of transformation rules generate the specific constructs for the target model (e.g., document or column-family).

Regarding performance, [8] benchmarks PostgreSQL and MongoDB. An OLAP-like workload is evaluated in both systems on real-world data from Github. The benchmark concludes that PostgreSQL yields higher performance results, but different design alternatives are not explored. [11] explores the impact of normalized collections w.r.t. embedded objects in MongoDB, and empirically shows that querying embedded objects is orders of magnitude faster than their normalized counterpart using joins. Similarly, [15] benchmarks systems in the NOSQL realm (i.e., MongoDB and CouchDB) as well as RDBMSs with built-in JSON support (i.e., PostgreSQL and MySQL). This work differs from our setting, as it focuses on CRUD transactions for a simple document structure.

3 Representational Differences

The term semi-structured describes data that have some structure but is neither regular nor known a priori [1]. For example, a JSON document consists of a nested hierarchy of key-value pairs with a single root. Child documents are an unordered sequence list of pairs with optional presence. Hence, JSON documents are self-descriptive, and do not require a schema declaration, despite a known structure facilitates storage and encourages queries [2]. Conversely, a structured database distinguishes schema and instances. The former is a set of attributes, each with a concrete domain, while the later is a tuple of values that belong to the corresponding domain in the previously declared schema. Hence, here, we present representational differences between semi-structured and structured data (i.e., equivalent alternatives to represent some datum exploiting the characteristics offered by each of both models), and discuss their potential impact on storage size, data insertion, and query performance. For each representational difference, we present patterns used in the empirical validation in Sect. 4.

3.1 Schema Variability

A common schema is defined for all instances in structured databases, but in JSON, there may exist potentially different document schemata inside the same collection. Here, we focus on comparing alternative ways to represent the schema.

3.1.1 Metadata Representation

Representing different schemata across JSON documents entails embedding their metadata into each instance (Fig. 1). This clearly impacts negatively the size of the database and consequently query performance. The more attributes are present, the more metadata (i.e., attribute names) will be embedded into each document. Additionally, the ratio between the size of data and metadata is clearly an important factor to consider (i.e., attribute name length w.r.t. its values). Thus, we need to consider (a) the absolute amount of metadata by analysing different number of attributes (from 1 to n), and (b) the relative amount of metadata by analysing different ratios (by increasing the value length from 1 to m, while at the same time that decreases the attribute name length in the same number of characters).

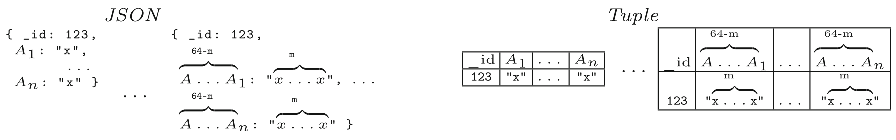

JSON
{ _id: 123, A_1: "x", ... A_n: "x" }
...
{ _id: 123, $\overbrace{A\ldots A_1}^{64-m}$: "$\overbrace{x\ldots x}^{m}$", ... $\overbrace{A\ldots A_n}^{64-m}$: "$\overbrace{x\ldots x}^{m}$" }
Tuple
_id | A_1 | ... | A_n
123 | "x" | ... | "x"
...
_id | $\overbrace{A\ldots A_1}^{64-m}$ | ... | $\overbrace{A\ldots A_n}^{64-m}$
123 | "$\overbrace{x\ldots x}^{m}$" | ... | "$\overbrace{x\ldots x}^{m}$"

Fig. 1. Alternative representations of Metadata

3.1.2 Attribute Optionality

Another feature of the semi-structured data model is the possibility to skip the representation on an attribute in the absence of its value (as the case of *J-Abs*, Fig. 2). However, it also supports to, either use a special value outside the domain (as in *J-NULL*) or use a specific value inside the attribute domain (as in *J-666*). Notice that in a relational representation, as the schema is fixed and common to all instances, only the last two options are possible (as in *T-NULL* and *T-666*, respectively). The impact on space and performance of these representations will vary depending on the percentage of absent/present values for the attribute.

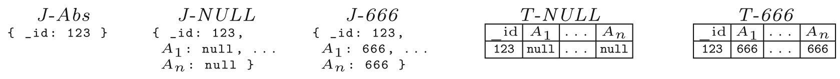

J-Abs
{ _id: 123 }
J-NULL
{ _id: 123, A_1: null, ... A_n: null }
J-666
{ _id: 123, A_1: 666, ... A_n: 666 }
T-NULL
_id | A_1 | ... | A_n
123 | null | ... | null
T-666
_id | A_1 | ... | A_n
123 | 666 | ... | 666

Fig. 2. Alternative representations for optional attributes

3.2 Schema Declaration

In order to benefit from Schema declaration and validation in semi-structured databases, one must adopt additional constructs. `JSONSchema` is a JSON-based schema language that allows to constrain the shape, types and values of JSON documents. Here, we will evaluate the impact of both structure plus data type declaration, and integrity constraint (IC) validation separately.

3.2.1 Structure and Data Types

To validate structure and data types, `JSONSchema` uses the `properties` key. For each attribute, it is possible to specify its data type, which can be either a primitive or complex object. Furthermore, the `required` key represents an array enumerating the list of expected attributes. Figure 3 depicts the exemplary document patterns considered. Clearly, this declaration has no impact on database size, since it does not grow with instances. However, it has a cost on insertion, corresponding to validating presence and domain, and on the other hand, it could potentially benefit query time by saving an explicit casting and type conversion.

J-Typ

```
{ _id: 123,        { "type": "object", "properties": {
  A1: k, ...         "A1": {"type": "number"}, ...
  A64: k }           "An": {"type": "number"},
                      required: ["A1",...,"An"] } }
```

T-Typ

_id	A_1	...	A_{64}
123	k	...	k

Fig. 3. Alternative representations of structure and data type validation

J-IC

```
{ _id: 123,      { "type": "object", "properties": {
  A1: k,           "A1": {
        ...          "type": "number",
  A64: k }           "minimum":-k',"maximum: k'}, ...
                   "An": {
                     "type": "number",
                     "minimum":-k',"maximum: k'} }
```

T-IC

_id	A_1	...	A_{64}
123	k	...	k

```
ALTER TABLE T ADD CONSTRAINT
val_A1 CHECK
(A1 BETWEEN -k' AND k');
...
ALTER TABLE T ADD CONSTRAINT
val_An CHECK
(An BETWEEN -k' AND k');
```

Fig. 4. Alternative representations of Integrity Constraints (IC) validation

3.2.2 Integrity Constraints

Besides the data type validation mechanisms, `JSONSchema` also offers means to represent integrity constraints for attributes. Here, as depicted in Fig. 4, we focus on enforcing ranges of values. In relational databases, this is achieved via `CHECK` constraints. As above, this has no impact on the size of the database but will have some on the insertion since it has to be checked before accepting the data. Despite this, it might also be used to perform some semantic optimization at query time; we consider this is technology-specific (i.e., not directly dependent on the data representation) and will not be evaluated in Sect. 4.

3.3 Structure Complexity

An RDBMS conforms to 1NF, yet a semi-structured one relaxes such restriction, which allows storing nested and multi-valued data. Here, we study the impact of different complexity degrees on data according to that.

3.3.1 Nested Structures

Documents allow to explicit into a data structure conceptually independent objects, which are accessed using dot notation. Yet, it is unclear what is the impact regarding size (i.e., with an increasing number of brackets in the document), and on querying such structures. To explore this, we will experiment with a range of levels and attributes (*Nest-one* and *Nest-all* in Fig. 5). Precisely, we will evaluate (a) increasing document sizes (i.e., *Nest-one*), and (b) constant document sizes (i.e., *Nest-all*); both w.r.t. the number of nesting levels. *Nest-1* indicates that there is only one attribute in the lowest level, while *Nest-all* contains less attributes the more levels we have. For instance, with 32 nesting levels, *Nest-one* has only A_{33}, while *Nest-all* has attributes A_{33} to A_{64}. Thus, in the latter, for every level we add together with the required extra characters (i.e., :, {, and }), we remove an attribute. Consequently, the overall size remains constant in terms of document length, but not in physical storage space due to the encoding of integer values being used.

Nest-one

```
{ _id: 123,
  L1: { ...
    Ln: {
      An+1: k }
            ...}}
```

Nest-all

```
{ _id: 123,
  L1: { ...
    Ln: {
      An+1: k, ...
      A64: k }...}}
```

J-Arr

```
{ _id: 123,
  A: [1,...,n]
        }
```

J-Att

```
{  _id:
        123,
  A1: k,
        ...
  An: k }
```

T-Arr

_id	A
123	[1,...,n]

T-Att

_id	A_1	...	A_n
123	k	...	k

Fig. 5. Representations of nesting structures and multi-valued attributes

3.3.2 Multi-valued Attributes

Only modern object-relational DBMSs have adopted variable-length multidimensional arrays as data type, an aspect present in JSON by definition. Yet, it is unclear what is the impact of managing such types. On bounded arrays, one could argue that it might be better to store each position as an independent attribute, as depicted in Fig. 5, where we distinguish, for both JSON and tuples, *array* and *multi-attribute* alternatives. Multi-valued attributes could also be stored in a separate normalized table, however such independent structure would compete for resources, heavily impacting insertion and query [10]. We consider such eviction policies are technology-specific, thus they will not be evaluated.

4 Experimental Evaluation

We conducted experiments to evaluate the choices discussed in Sect. 3, using PostgreSQL v12 (which supports native JSON storage) to compare the differences between relational and JSON alternatives. We also used MongoDB v4.2 (nowadays, the most popular document store) to validate the consistency of results. Note our objective is not to perform a technological comparison, but to evaluate the impact of document design choices. No specific tuning was performed for any system, using the default parameters. We disabled compression in MongoDB to facilitate its comparison with PostgreSQL, and cleared the operating system cache and restarted the DBMS between each execution to clear caches. We got three metrics: (a) storage size in MB; (b) overall runtime of insertions in seconds; and (c) median runtime to aggregate a numeric attribute in seconds over 20 repetitions. To store JSON in PostgreSQL, we created a table with two attributes: a `CHAR(24)` to store the ID (equivalent to `Object_ID` in MongoDB) and a `JSONB` to store the document. Then, we generated 1 million random documents according to each schema pattern in Sect. 3, over an exponentially increasing parameter, which were inserted in 100 batches of 10 K documents. Due to space limits, we omit the individual figures[2]. To minimise impedance mismatch, queries return a single value aggregating numerical attributes. Note that MongoDB stores 32-bits integers[3], while PostgreSQL uses 64-bits[4], which in the end causes differences on storage size and consequently in insertion and query performance.

[2] Source code and all graphs available at https://github.com/dtim-upc/MongoDBTests.

[3] https://docs.mongodb.com/manual/reference/bson-types.

[4] https://www.postgresql.org/docs/12/datatype-json.html.

4.1 Schema Variability

For schema variability, we conducted three experiments overall because we already had two patterns regarding metadata embedding (Sect. 3.1.1): (i) change the number of numeric attributes in a document; and (ii) change the data-metadata ratio, keeping a fixed number of attributes.

Varying Document Size. According to our experiments JSON always requires more space than tuples, due to metadata being replicated in every document. We can observe the same trend in insertion times. However, although storage space for a tuple is smaller in all cases, insertion time is shorter only for few (i.e., four) attributes. Beyond that, JSON insertion is faster (due to no type checking, as shown later in Sect. 4.2). At query time the runtime increases with the number of attributes. However, oppositely to insertion, tuples perform faster (since they benefit from the work done at insertion time). In all cases, we can see that PostgreSQL and MongoDB follow the same trend on storing JSON. They only differ in the physical format, which requires less space in the latter (64-bit vs. 32-bit integers). Thus, MongoDB generates less I/O (roughly half), improving insertion and query time.

Constant Document Size. Aiming to stabilise the overall size of the document, we keep constant the sum of characters between attribute name and value. Thus, we have one numerical attribute for the queries and consider nine other string attributes, changing at once their data to metadata ratio by changing the length of attribute name and value keeping a constant of 64 characters for both together. The number is chosen based on PostgreSQL having a limit of 63 characters for attribute names, so the attribute name length ranges from 1 to 63 and the value length from 63 to 1. Since attribute name is only stored once, independently of the number of tuples, the storage space taken by the tuples decreases with the growth of the attribute name length. Oppositely, attribute names are redundantly stored in all documents in JSON, so the overall size remains constant except for 63 characters, seemingly due to the presence of a step function in physical storage allocation. This is confirmed in MongoDB, where the gradual growth in space is more apparent. Interestingly, PostgreSQL and MongoDB storage size for JSON is much closer in this experiment as most of the attributes are strings instead of integers. Insertion and query times follow the same trend as the attribute length grows indicating I/O is always the dominant factor.

Optional Attributes. Regarding attribute optionality, we consider five alternatives to represent the absence of values in the attributes (Sect. 3.1.2). Thus, the pattern consists of 64 integer attributes (potentially removed all at once), and one fixed-length string of size 64 to guarantee a minimal document size when the former are removed. Thus, we varied the percentage of documents without value for their integer attributes. Regarding storage space, the worst option to represent absence of data is using a value inside the domain (i.e., *T-666* and *J-666*), which keeps a constant size. In both tuples and JSON, we can use a *null* special value (i.e., *T-NULL* and *J-NULL*), which clearly saves space as attribute values disappear. However, the complete absence of the attribute in

JSON (namely *J-Abs*), reduces the storage space the most due to the saving also in the metadata. As before, storage space in MongoDB follows the same trend as in PostgreSQL, but with smaller values due to the different encoding of integers. Regarding insertion time, the trend coincides with that of the storage used for JSON in both systems. However, tuples in PostgreSQL keep a constant insertion time, because the dominant factor is not I/O, but validation and formatting of data, which is not even compensated by the saving in metadata storage. When querying the data, we tested both summing and counting their presence with similar results. In all cases, the dominant factor of the query time is I/O, and consequently follows the trends and proportions of storage space.

4.2 Schema Declaration

As discussed, schema declaration does neither affect the overall storage size nor query time. Thus, we measure insertion time for both data types and ICs.

Type and Constraint Validation. Regarding type and IC checking (Sects. 3.2.1 and 3.2.2), we generated documents with 64 attributes and declared type and ICs in an incremental manner (from 1 to 64). To enforce JSON schema declaration in PostgreSQL, we used the *postgres-json-schema*[5] extension. In MongoDB, this is a built-in feature that can be simply enabled with the operator `$jsonSchema`, which is provided at creation time of the collection. In tuples, all data types must always be declared, leading to constant insertion time. Oppositely, when inserting JSON, time increases with data types declaration, confirming the consequent overhead. Checking concrete ICs on top of data types, substantially increases the overhead. Both systems confirm trends, the only difference being that built-in mechanism of MongoDB being faster.

4.3 Structure Complexity

Finally, we analyse the impact of breaking first normal form by either nesting documents (Sect. 3.3.1) or storing multi-valued attributes (Sect. 3.3.2). Notice that only the latter is available in relational implementations.

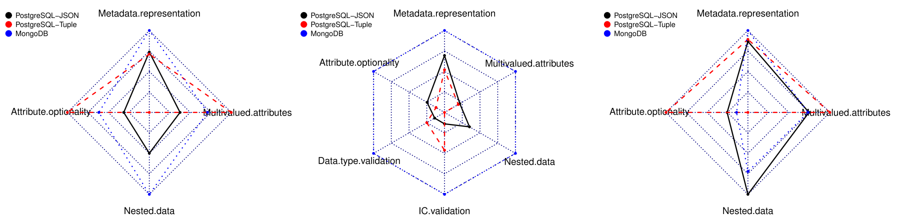
Nested Structures. The storage size of nesting one attribute increases the document size with the increasing number of levels. MongoDB slightly increases the physical storage when the number of levels increases, even with constant document size. The integer encoding difference (64-bits vs. 32-bits) explains this opposite behavior. The insertion time follows the same trend of the storage size. We noticed an extra overhead in MongoDB beyond that of purely I/O. PostgreSQL performs better than MongoDB (despite having higher I/O), and MongoDB have a clear upward trend with the increasing number nesting levels as opposed to constant runtime in PostgreSQL confirms the overhead nesting generates in MongoDB.

[5] https://github.com/gavinwahl/postgres-json-schema.

Multi-valued Attributes. Regarding the storage of multi-valued attributes (Sect. 3.3.2), we generated documents with the number of values per attribute ranging from 2 to 64 for the different options. For tuples, we used either PostgreSQL native array storage or separate attributes for each value, and similarly for JSON either as an array in the document, or as separate attributes. Regarding storage size, both systems take more space for JSON than tuples, because of the saving of tuples on metadata replication. While in tuples both options use the same space, in JSON arrays are clearly more efficient, since separate attributes require more characters (the same behavior is confirmed in MongoDB, but mitigated by its smaller encoding of integers). Despite insertion time in JSON is dominated by I/O, in tuples inserting to an array is faster than inserting multiple attributes, due to the overhead of parsing and validating independent attributes in front of one single array. Nevertheless, the extra processing at insertion time pays off at query time, where processing the independent attributes is faster than digging inside the array. For JSON, we appreciate the same benefit of querying independent attributes in PostgreSQL, but surprisingly the opposite behavior in MongoDB, where processing the array is systematically faster. When summin indivudual attributes, MongoDB has a built-in function that sums the content of the array, which is more efficient, and on the contrary, PostgreSQL needs to unwind the array in order to calculate the sum, which is more expensive.

5 Discussion

Figure 6 summarizes all results with regard to storage space, load time, and query time. For this, we calculated the average of all measurements per representational difference for each of the three options (i.e., Tuples, and JSON in both systems). Since data follows different patterns in each case, we separately min-normalize per case (e.g., divide the minimum of the three averages for nested data by the average for Tuples) and plot them all in the corresponding radar chart. This means values further away from the center of the radar are better than the ones closer, and the bigger the area of the polygon, the better the system performs.



(a) Storage (b) Insertion time (c) Query time

Fig. 6. Multidimensional view of experimental results

According to Fig. 6a storing tuples takes the least amount of space in all cases except metadata representation. On interpreting this, we acknowledge the impact of the ratio between metadata and data, which is fixed to be relatively high in all experiments. Thus, attribute names should always be encoded in JSON to shorten them as much as possible and improve that ratio. Obviously, this is more relevant, for example, if values are numeric than if they are strings (the former requiring less space, in general). Within JSON, PostgreSQL storage size is much larger than MongoDB in all the cases, due to the different encoding of integers (64-bits vs. 32-bits).

According to Fig. 6b it is clear looking at PostgreSQL that loading JSON is faster than tuples, except for data type and integrity constraint validations. However, it is important to note that the validation of JSON was carried out through a third-party plugin, which definitely impacts the results. MongoDB being a native document store, has a clear advantage over PostgreSQL JSON storage in loading data (at the end of the day, JSON is stored as a column in a PostgreSQL table), beating even tuple storage in the validation dimensions. This, however, can come not only from using JSON format but from other DBMS characteristics (e.g., lack of ACID transactional support).

Finally, Fig. 6c depicts that tuples, in general, perform better in queries. This is so because they use less space, in general, and benefit from validation at insertion time. Thus, we can see that when the space-saving is lost depending on the data-metadata ratio, so the benefit is mostly lost at query time, as well. Nonetheless, JSON representation is at a disadvantage, as each of the documents needs to be parsed and processed on demand. Consequently, we should consider the trade-off between the pressure of fast ingestion and the long term benefit of recurring queries. It is also interesting to see that even though the storage size of JSON is larger in PostgreSQL, this is still faster than MongoDB. We believe this fact results from the differences in how query engines handle the calculations. PostgreSQL benefits here from the well-optimized aggregation operations in the relational engine, which data stored in JSON format also have access to.

6 Conclusions and Future Work

In this paper, we studied the impact of physical design choices for NOSQL databases according to six different characteristics. We conclude that there is no *ace of spades*, when designing JSON documents. However, we identified a crucial trade-off between insertion and query performance. Nowadays, organizations are shifting their data repositories to flexible representations following a *schema-on-read* approach, but we have empirically shown that such an approach might have several shortcomings in front of query-intensive workloads. As future work, we aim to extend our experiments taking into account more features from the DBMS in use. This involves considering caching mechanisms or indexing structures.

References

1. Abiteboul, S.: Querying semi-structured data. In: ICDT (1997)
2. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web - From Relations to Semistructured Data and XML. Morgan Kaufmann, Burlington (2000)
3. Ambler, S.: Agile Database Techniques: Effective Strategies for the Agile Software Developer. Wiley, Hoboken (2003)
4. Atzeni, P., Bugiotti, F., Cabibbo, L., Torlone, R.: Data modeling in the NoSQL world. Comput. Stand. Interfaces **67**, 103149 (2020)
5. Badia, A., Lemire, D.: A call to arms: revisiting database design. SIGMOD Rec. **40**(3), 61–69 (2011)
6. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM **13**(6), 377–387 (1970)
7. de la Vega, A., García-Saiz, D., Blanco, C., Zorrilla, M.E., Sánchez, P.: Mortadelo: automatic generation of NoSQL stores from platform-independent data models. Future Gener. Comput. Syst. **105**, 455–474 (2020)
8. Hernández, A., etal.: Performance Benchmark PostgreSQL/MongoDB (Technical report) (2019)
9. Herrero, V., Abelló, A., Romero, O.: NOSQL design for analytical workloads: variability matters. In: ER (2016)
10. Hewasinghage, M., Abelló, A., Varga, J., Zimányi, E.: DocDesign: cost-based database design for document stores. In: SSDBM (2020)
11. Kanade, A., Gopal, A., Kanade, S.: A study of normalization and embedding in MongoDB. In: IACC (2014)
12. Mohan, C.: History repeats itself: sensible and NonsenSQL aspects of the NoSQL hoopla. In: EDBT (2013)
13. Scherzinger, S., Sidortschuck, S.: An empirical study on the design and evolution of NoSQL database schemas. CoRR, abs/2003.00054 (2020)
14. Sadalage, P., Fowler, M.: NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley Professional, Boston (2012)
15. Truica, C., Radulescu, F., Boicea, A., Bucur, I.: Performance evaluation for CRUD operations in asynchronously replicated document oriented database. In: CSCS (2015)

Self-service Business Intelligence over On-Demand IoT Data: A New Design Methodology Based on Rapid Prototyping

Julian Eduardo Plazas[1,3](✉), Sandro Bimonte[2], Michel Schneider[3], Christophe de Vaulx[3], and Juan Carlos Corrales[1]

[1] GIT, Universidad del Cauca, Calle 5 No 4-70, 190003 Popayán, Cauca, Colombia
{jeplazas,jcorral}@unicauca.edu.co

[2] TSCF, INRAE Clermont-Ferrand, Université Clermont Auvergne, 63178 Aubière, France
sandro.bimonte@inrae.fr

[3] LIMOS, Campus des Cézeaux, Université Clermont Auvergne, 63178 Aubière, France
michel.schneider@isima.fr, christophe.de_vaulx@uca.fr

Abstract. Data Warehouse (DW) and OLAP systems are acknowledged as first citizens of Business Intelligence (BI) technologies, allowing the on-line analysis of huge volumes of data. However, traditional data-driven BI might not be enough to compete in the context of Industry 4.0, since the collection and analysis of data from the Internet of Things (IoT) requires a more responsive approach. Therefore, in this work, we present a new design methodology for Self-Service DW with On-Demand IoT Data, which is accompanied by a new UML profile for Stream Data Warehouses based on IoT data.

Keywords: Business Intelligence · Internet of Things · Conceptual modelling · Design methodology · UML profile

1 Introduction

The acquisition and analysis of Big Data coming from the Internet of Things (IoT) are key competitive aspects for any enterprise and organisation. Indeed, they allow decision-makers to be aware of the present and possible future situations [17]. However, properly acquiring, analysing and exploiting IoT-generated Big Data are complex tasks. Hence, different types of experts are involved in the design and use of such applications: IoT experts for the IoT implementation and data acquisition; Data scientists for Big Data Analytics and Business Intelligence (BI) implementation (*i.e.* BI experts in our case); and Domain experts and decision-makers (*i.e.* business users) for information exploitation [15]. Traditionally, business users interact only with the data scientist to define the required

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 84–93, 2020.
https://doi.org/10.1007/978-3-030-54623-6_8

analysis over already-existing and -deployed IoT data. IoT experts have little or no interaction with the other actors, and the IoT data is *by default*.

This approach is usually known as *data-driven*. Nevertheless, with the advent of IoT and Industry 4.0, data-driven BI seems inefficient and ineffective. The available data is not always appropriate for decision-making, which causes significant delays in the analysis of information [3,9]. Therefore, traditional data-driven BI is evolving in two main aspects: i) reducing the dependence on Information Technologies (IT) experts (IoT and BI) through *self-service BI* [9]. And ii) improving the pertinence of the acquired data considering the business requirements through *BI with on-demand data* [3].

Consequently, in this work, we propose a new vision *where the Big Data Analytics (BI) paradigm will shift from a data-driven approach to a requirement-driven one:* ***self-service BI with on-demand data*** (SSBI-ODD). In this paradigm, data are collected on-demand from the IoT, and BI analysis are easy to run according to the business needs. Thus, business users define both the collection and analysis without completely relying on IoT and BI experts. In this way, a design methodology for our vision of SSBI-ODD rises some new important open issues: (i) Few experts have knowledge of both IoT and BI systems. Thereby, it is necessary to constitute teams with different experts, which usually drives to increased development time and reduced capacity for meeting the original requirements [2,12]. And (ii) SSBI-ODD systems will be highly complex. So, even a conceptual model approach [8] for SSBI-ODD system must provide support for IoT and stream or classical BI data at the same time, and thus it could represent a challenge for anyone trying to understand it as a whole.

To address these issues, we base our *new design methodology for SSBI-ODD* on the ProtOLAP methodology [3]. In particular, this methodology focuses on Data Warehouse (DW) and OLAP systems, the two main technologies of BI that analyse huge volumes of multidimensional data. ProtOLAP follows the main DW design principles: high involvement of business users, focus on conceptual modelling and rapid prototyping [8]. Consequently, our new design methodology allows all actors (BI and IoT experts, and business users) to discuss and define the analysis requirements for classical and on-demand IoT data, and develop prototypes rapidly and automatically for evaluation and deployment. Moreover, we investigate different possible scenarios to achieve the integration of IoT and classical data at the conceptual level. In particular, we propose a *new UML profile for Stream Data Warehouses (SDW)* representing the IoT data as SDW facts.

2 Related Work

Different authors have proposed model-driven methodologies and architectures for the development of IoT applications. All of these works consider the importance of using IoT-generated data into Big Data Analytics systems to extract value. For example, [1] defines an architecture for integrating IoT data into Big Data systems for advanced manufacturing. [11] defines a methodology for the design of smart IoT-based applications, separating the concepts and roles for processes, semantics and Big Data management. And [13] defines a methodology for developing IoT applications considering four different roles besides the domain experts. However, to the best of our knowledge, most of these methodologies focus only on the IoT part of the system, without providing clear steps or models for the data integration.

Moreover, regarding conceptual models for BI and their automatic implementation, some works propose to use model-driven approaches in the context of Data Warehouse [16], data mining [7], and spatial analytics, amongst other topics. However, to the best of our knowledge, only [2] defines a UML profile for representing BI indicators focusing on stream data for DW. Nevertheless, this UML profile does not represent how the stream is generated.

Regarding IoT (*i.e.* the stream generation part), we have found no evidence of approaches providing rapid prototyping tools from a data representation. Firstly, works proposing meta-models for representing IoT data cannot help in the firmware development process [10,14]. Secondly, several works proposes different approaches for generating firmware from models [5,6,12,13], yet such models have little or no focus on data.

Consequently, to the best of our knowledge, existing works do not consider the multidimensional analysis of on-demand IoT data, neither they allow for a complete modelling and development process of IoT-based BI applications. We thereby conclude that there are no conceptual models or methodologies enabling SSBI-ODD, which constitutes our proposal in a first approach on the subject.

3 Our Methodology

In this section, we present our methodology for SSBI-ODD, focusing on DW and IoT (Fig. 1), which is an extension of the ProtOLAP methodology [3]. ProtOLAP is a methodology that allows for rapid prototyping of DW. It is mainly based on: (i) the automatic implementation of DW schema models from UML models; (ii) validation of decision-makers requirements via visualization of simulated warehoused data by means of OLAP clients (right part of Fig. 1). Full details of ProtOLAP are thoroughly explained in [3]. In particular, we have added a new set of steps in the methodology that allows for the definition of the IoT data acquisition system. In this way, it complements the goal of [3] with the direct integration of a different and complex data source. Figure 1 represents the steps from ProtOLAP that remain the same with an orange frame and the new steps with a green frame.

Our methodology (Fig. 1) is composed by six phases:

In the **Requirement Elicitation** step, the Business Users state their analysis needs in natural language (*i.e.* an informal definition). Throughout the discussion with the BI and IoT experts, these analysis needs are defined in terms of *indicators* (*e.g.* measures and aggregations). This step is quite similar to one proposed in [3], but here the business users must also express their needs in terms of data collected by the IoT (*i.e.* what, when, and where).

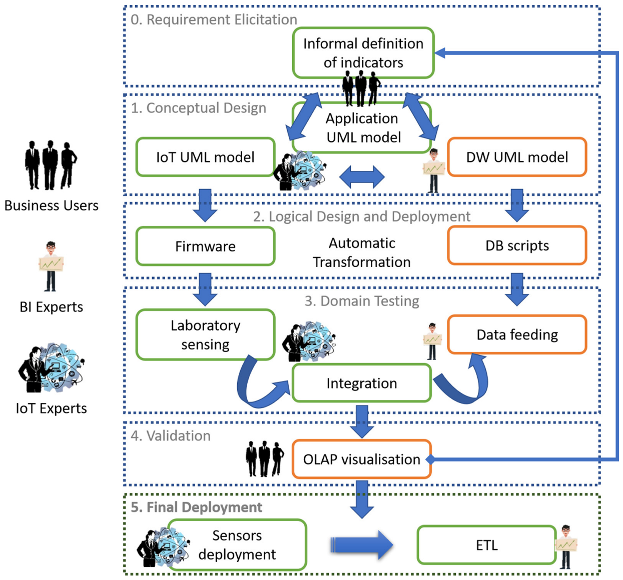


Fig. 1. Methodology for self-service DW over on-demand IoT data. (Color figure online)

Then, in the **Conceptual Design** step, the experts in both IoT and BI areas define the best way to formalise the *indicators* as an *Application UML model*. The IoT and BI experts can then use this model to explain the technological solution to the business users, discuss with them, and refine the application design. Once the *Application UML model* is complete and appropriate enough for the three roles (*i.e.* IoT and BI experts, and business users), the IoT experts start working on the data-centric design of the IoT part (*IoT UML model*), while the BI experts work on the data-centric design of the BI part (*DW UML model*) as defined in [3]. Both models have to be compliant and integrated in only one common model.

In the **Logical Design and Deployment** step, the models for the IoT part (*IoT UML model*) and the BI part (*DW UML model*) are automatically transformed into *Firmware* and *Database (DB) scripts* respectively. The *Firmware* allows programming the IoT devices, while the *DB scripts* define the relational schema and metadata of the DW as described in [3]. In this phase, automatic transformation from the conceptual models is a key feature that significantly reduce the implementation effort allowing for rapid prototyping. Therefore, we suggest the use of model-driven approaches such as [4], though different approaches might also be valid.

For the next step, **Domain Testing**, IoT and BI experts work together again. Firstly, the IoT experts prepare some IoT devices for gathering data in a test environment (*Laboratory sensing*). Secondly, the BI experts prepare an initial deployment of the DW capable of receiving data (*Data feeding*) as defined in [3]. Finally, they integrate the two experimental subsystems (*Integration*).

This integration sub-step is done manually since it depends on the particular application needs as discussed in Sect. 5. In this phase, IoT and BI experts must verify the appropriate data collection, delivery, reception and analysis; an thus it outputs an operative prototype of the IoT-based BI system.

In the **Validation** step, the business users check the prototype of the system with a common *OLAP visualisation* tool. By exploring the data and analysis provided by such prototype, they can validate if the system (and thus the underlying conceptual model) correctly provides the defined *indicators*. If the prototype is not appropriate, the business users return to the discussion with the experts on IoT and BI, further refining the *Application UML model*.

Otherwise, if the prototype is deemed appropriate, the process advances to the sixth and final phase: **Final deployment**. This phase consists in deploying the sensor devices and finalising the ETL procedures to receive and load their data into the DW.

The following sections provide deeper details and examples of the most important (early) steps of our methodology, which enable SSBI-ODD. Specifically, Sect. 4 provide insights in the *IoT UML Model* definition and the consequent automatic generation of *IoT Firmware*; while Sect. 5 describes three different scenarios for *integrating IoT and DW models* and systems.

4 Conceptual Design and Logical Design and Deployment of IoT Data

In this section, we describe the conceptual model for the IoT part and its automatic implementation. The data-centric UML profile for IoT (Fig. 2-A) allows the IoT and BI experts to define the IoT in terms of the required data. The instantiated data models (*e.g.* Fig. 2-B) are very simple and readable, and are thereby useful for discussing with the business users; enabling an effective refinement process.

This profile (Fig. 2-A) is composed by six Stereotypes, four Tagged Values and two Enumeration data types. The main Stereotype is `Sensor_DataStream`, which describes the data that is available from the IoT in a Class. The *`Sensed_WeatherConditions`* Class in Fig. 2-B applies this Stereotype. This Class is contained into the *`Sensor`* Package, which applies the `Sensor` Stereotype.

Besides, the `Sensor_Variable` Property Stereotype describes the variables that the IoT senses and delivers without any computation, such as *`timeStamp`* in the example. Moreover, the `Sensor_AggregatedVariable` Property Stereotype describes those variables that have a previous temporal aggregation inside the IoT, like *`TemperatureSum`* and *`AvgTemperature`* in the example. Besides, this Stereotype defines the `Aggregation` Tag related to the `Sensor_Aggregation` Enumeration in order to state the aggregation operation affecting each variable.

Finally, the `Sensor_TupleDelivery` and `Sensor_TimeDelivery` Operation Stereotypes describe how the IoT operates and delivers its data to the BI system. Firstly, the `Period` defines how often IoT is delivering the sensed data to the BI. This `Period` represents a number of tuples or samples in `Sensor_TupleDelivery`,

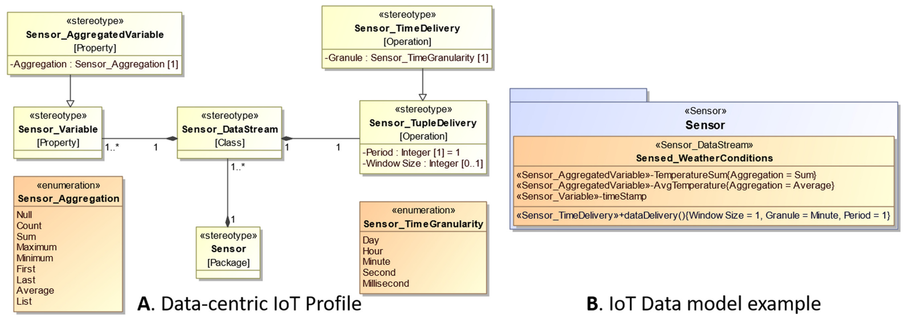
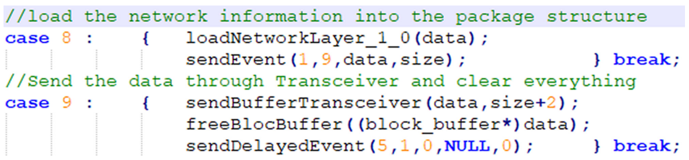

«stereotype»
Sensor_AggregatedVariable
[Property]
-Aggregation : Sensor_Aggregation [1]
«stereotype»
Sensor_TimeDelivery
[Operation]
-Granule : Sensor_TimeGranularity [1]
«stereotype»
Sensor_Variable
[Property]
1..*
1
«stereotype»
Sensor_DataStream
[Class]
1
1
«stereotype»
Sensor_TupleDelivery
[Operation]
-Period : Integer [1] = 1
-Window Size : Integer [0..1]
1..*
1
«enumeration»
Sensor_Aggregation
Null
Count
Sum
Maximum
Minimum
First
Last
Average
List
«stereotype»
Sensor
[Package]
«enumeration»
Sensor_TimeGranularity
Day
Hour
Minute
Second
Millisecond
A. Data-centric IoT Profile
«Sensor»
Sensor
«Sensor_DataStream»
Sensed_WeatherConditions
«Sensor_AggregatedVariable»-TemperatureSum{Aggregation = Sum}
«Sensor_AggregatedVariable»-AvgTemperature{Aggregation = Average}
«Sensor_Variable»-timeStamp
«Sensor_TimeDelivery»+dataDelivery(){Window Size = 1, Granule = Minute, Period = 1}
B. IoT Data model example

Fig. 2. Data-centric UML profile for IoT (A) and data model example (B).



```
//load the network information into the package structure
case 8 :     {    loadNetworkLayer_1_0(data);
                  sendEvent(1,9,data,size);            } break;
//Send the data through Transceiver and clear everything
case 9 :     {    sendBufferTransceiver(data,size+2);
                  freeBlocBuffer((block_buffer*)data);
                  sendDelayedEvent(5,1,0,NULL,0);      } break;
```

Fig. 3. Fragment of example firmware code generated with the automatic transformation tool.

and an amount of time in `Sensor_TimeDelivery`, which also includes the `Granule` Tag in relation with the `Sensor_TimeGranularity` Enumeration to set an unit for time. Secondly, `WindowSize` represents the amount of data aggregated before delivering. In `Sensor_TupleDelivery` it represents the number of tuples or samples operated, while in `Sensor_Time-Delivery` it represents a time window before the delivery that is also associated to the `Granule`. For example, the IoT represented in Fig. 2-B delivers its data every 1 min, and calculates the aggregates (*i.e. Sum* and *Average*) with the data collected during that minute.

After the data model of IoT is complete, refined, and agreed by all the roles, the IoT experts can proceed to the Logical Design and Deployment. In this phase, the IoT experts must select an implementation IoT device, and define some physical features of the application: device address, device identifier, sensing probes, sensing period, internal data operations and delivery interface.

Once the IoT experts set these features, the IoT experts can use an automatic model-to-code transformation tool. Such a tool takes as input the XMI (XML Metadata Interchange) file of the IoT data model, the physical features set by the IoT experts and the characteristics of the selected IoT device in order to generate the firmware. For example, Fig. 3 shows a fragment of firmware expressed in C code for delivering the IoT data through the *Transceiver* interface of the device. This firmware is then compiled and loaded into the IoT hardware, which starts providing the required data.

5 Conceptual Integration of IoT Data in BI

After both the IoT and the DW are implemented in their first separated prototype (Domain Testing phase), their integration should be as simple as defining a gateway that connects the output interface of the IoT and the input interface of the BI. The data acquisition (IoT) and reception (DW) processes must be compatible *from the design of the system.* Thereby, in this section, we describe three different approaches for providing a unified and complete data model of IoT-based BI applications. These approaches corresponds to different analysis needs expressed by business users. We present it using the agriculture example introduced in the previous sections.

The **naive approach** considers that: the IoT and the DW are represented in two different data models; the DW Fact has equivalent attributes with the IoT `Sensor_DataStream` (*i.e.* there is no need for ETL or any transformation on the IoT data); and the two models can be related through a one-to-many association. *This approach allows OLAP analysis over IoT data that are permanently stored in the DW without requiring any transformation.*

The **ETL approach** is similar than the Naive one, it still considers that the IoT and the DW are represented in two different data models and the two models can be related through a one-to-many association. However, the attributes in the DW Fact are different than that of the IoT `Sensor_DataStream` and thus they require a transformation process (*i.e.* ETL) before being loaded into the DW. *This approach allows OLAP analysis over IoT data that are permanently stored in the DW with some previous transformations that cannot be achieved directly inside the IoT.* Indeed, sometimes sensors have not enough computational or battery resources to provide any kind of transformation. Therefore, they must be executed outside of the IoT.

Recent works propose to provide OLAP queries over data streams. Leveraging this interesting idea, we can use our IoT UML profile to represent the stream data-source of a **Stream DW**, allowing for a continuous OLAP analysis over streamed IoT data without requiring additional transformations or permanent storage. This approach is different from existing ones since it models the whole IoT-based BI application in a single model using a single profile. More precisely, we propose to use the UML profile for Stream Data Warehouses (SDW) of [2]. This work extends the [4] profile for classical data with a `Stream Fact` and a `Window Dimension`. The `Stream Fact` allows representing temporally-constrained data (*i.e.* stream), while the `Window Dimension` defines for how long the SDW will keep the data to run the required analysis.

Figure 4 shows how we have further extended the [2] profile to use our `Sensor_DataStream` of IoT as a `StreamFact` for SDW. Our extensions are highlighted with blue dotted frames. Firstly, Fig. 4-A shows the resumed SDW profile for the `HypercubeStream` Package, which replaced the classical `Hypercube` Package. To extend this part of the profile, we define the `AbstractStreamFact` Abstract Class Stereotype as a generalisation of `StreamFact`, associating it to the `HypercubeStream` and the `TimestampAggLevel`. Then, we add our `Sensor_DataStream` as one of the specifications of `AbstractStreamFact`, which

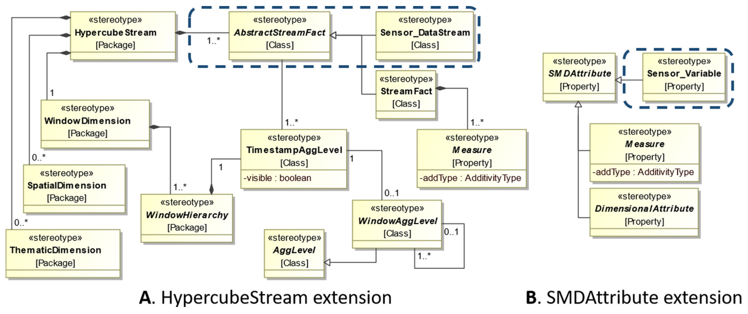

«stereotype» HypercubeStream [Package]
«stereotype» AbstractStreamFact [Class]
«stereotype» Sensor_DataStream [Class]
«stereotype» StreamFact [Class]
«stereotype» WindowDimension [Package]
«stereotype» SpatialDimension [Package]
«stereotype» ThematicDimension [Package]
«stereotype» WindowHierarchy [Package]
«stereotype» TimestampAggLevel [Class] -visible : boolean
«stereotype» Measure [Property] -addType : AdditivityType
«stereotype» WindowAggLevel [Class]
«stereotype» AggLevel [Class]
«stereotype» SMDAttribute [Property]
«stereotype» Sensor_Variable [Property]
«stereotype» DimensionalAttribute [Property]
A. HypercubeStream extension
B. SMDAttribute extension

Fig. 4. Extension of the Stream Data Warehouse profile.

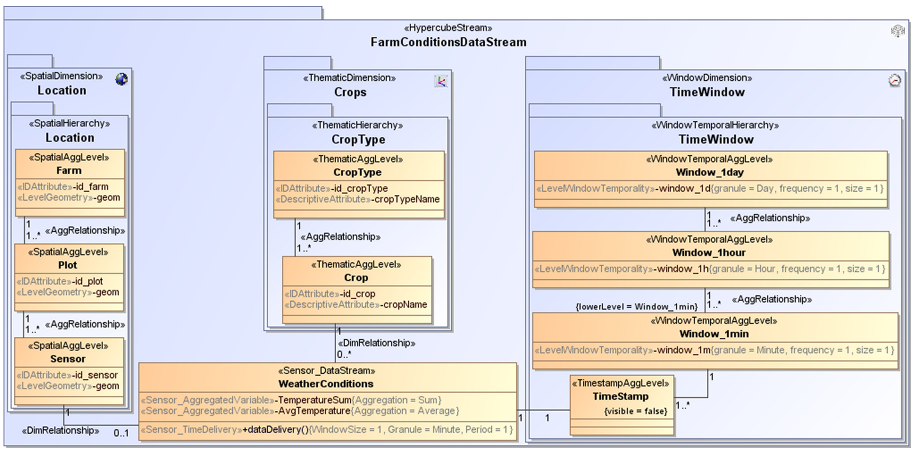

«HypercubeStream» FarmConditionsDataStream
«SpatialDimension» Location
«SpatialHierarchy» Location
«SpatialAggLevel» Farm «IDAttribute»-id_farm «LevelGeometry»-geom
«SpatialAggLevel» Plot «IDAttribute»-id_plot «LevelGeometry»-geom
«SpatialAggLevel» Sensor «IDAttribute»-id_sensor «LevelGeometry»-geom
«AggRelationship»
«DimRelationship»
«ThematicDimension» Crops
«ThematicHierarchy» CropType
«ThematicAggLevel» CropType «IDAttribute»-id_cropType «DescriptiveAttribute»-cropTypeName
«ThematicAggLevel» Crop «IDAttribute»-id_crop «DescriptiveAttribute»-cropName
«WindowDimension» TimeWindow
«WindowTemporalHierarchy» TimeWindow
«WindowTemporalAggLevel» Window_1day «LevelWindowTemporality»-window_1d{granule = Day, frequency = 1, size = 1}
«WindowTemporalAggLevel» Window_1hour «LevelWindowTemporality»-window_1h{granule = Hour, frequency = 1, size = 1}
{lowerLevel = Window_1min}
«WindowTemporalAggLevel» Window_1min «LevelWindowTemporality»-window_1m{granule = Minute, frequency = 1, size = 1}
«Sensor_DataStream» WeatherConditions
«Sensor_AggregatedVariable»-TemperatureSum{Aggregation = Sum}
«Sensor_AggregatedVariable»-AvgTemperature{Aggregation = Average}
«Sensor_TimeDelivery»+dataDelivery(){WindowSize = 1, Granule = Minute, Period = 1}
«TimestampAggLevel» TimeStamp {visible = false}

Fig. 5. Example data model of a Stream Data Warehouse using IoT data.

allows us to analyse IoT data in the different dimensions. Secondly, Fig. 4-B shows the relationship between the `Measures` of the `StreamFact` (*e.g.* `NumericalMeasure`) and the `Sensor_Variable` of the `Sensor_DataStream`. In this part of the profile, we add our `Sensor_Variable` (and thus the `Sensor_AggregatedVariable`) as one of the specifications of `SMDAttribute`, the generalisation of `Measure`. Note that these representations (Fig. 4) are resumed, *i.e.* do not exhibit all the depth of the profiles. Therefore, their understanding must be complemented with the original SDW profile [2] and the IoT profile (Sec. 4). Nevertheless, we provide an example in the agriculture domain using our extended profile for IoT-based BI applications (Fig. 5).

This model (Fig. 5) presents a `HypercubeStream` named *`FarmConditionsDataStream`* to analyse the weather conditions on a farm over different *`Time Windows`* regarding the *`Crops`* and the *`Location`* (*i.e.* dimensions). The observation *`Time Windows`* can last one minute, one hour or one day. The *`Crops`* can be analysed by each individual *`Crop`* or grouped by *`CropType`*. Besides, data can also be analysed at the level of `Sensors`, `Plots`, or for the whole `Farm`. Furthermore, the `Weather Conditions` are sensed through an IoT device, which every minute loads into the SDW the accumulated (`TemperatureSum`) and average (`AvgTemperature`) air temperature values from the last minute. In this way, this model could answer queries such as: *"Which Plot is currently presenting the hourly minimum average temperature?"* Or "*List all the plots with avocado with a total accumulated temperature under 500 during this hour*".

6 Conclusion and Future Works

In the context of the fourth industrial revolution, enterprises face new challenges that decision-support technologies like BI must help to overcome. However, the collection and analysis of classical data are no longer enough for effective decision-making. Indeed, these technologies must evolve to collect and analyse contextual data from the IoT. We have thereby presented our vision for SSBI-ODD, which empower the business users to define the most relevant data and analysis needs according to the business goals, and allows supplying such needs with reduced effort from IT experts. We support our vision with a new design methodology and a UML profile for self-service DW over on-demand IoT data. Our proposal exhibits two main principles: (i) the participation of all the roles in the design process through formal yet readable models. (ii) Rapid prototyping based on model-driven approaches and model-transformation tools. Consequently, it allows efficiently developing effective SSBI-ODD applications for improved decision-making. Finally, our on-going works are the evaluation of our methodology in a real scenario, the extension of our IoT data profile with IoT-specific ETL functions, and the definition of a rapid-prototyping approach for the gateway between the IoT and the DW.

Acknowledgements. This work was partially supported by French ANR projects VGI4Bio (ANR-17-CE04-0012) and "Investissements d' Avenir" through the IDEX-ISITE initiative CAP 20–25 (ANR-16-IDEX-0001), and the Colombian project IoT-Agro of Universidad del Cauca (VRI ID4633). We also thank COLCIENCIAS (Colombia) for the PhD scholarship granted to Julián Eduardo Plazas.

References

1. Arantes, M., Bonnard, R., Mattei, A.P., De Saqui-Sannes, P.: General architecture for data analysis in industry 4.0 using SYSML and model based system engineering. In: 2018 annual IEEE international systems conference (SysCon), pp. 1–6. IEEE (2018)
2. Bimonte, S., Boussaid, O., Schneider, M., Ruelle, F.: Design and implementation of active stream data warehouses. Int. J. Data Warehouse. Min. (IJDWM) **15**(2), 1–21 (2019)
3. Bimonte, S., Edoh-Alove, É., Nazih, H., Kang, M.A., Rizzi, S.: Protolap: rapid olap prototyping with on-demand data supply. In: Proceedings of the sixteenth international workshop on Data warehousing and OLAP, pp. 61–66 (2013)

4. Boulil, K., Bimonte, S., Pinet, F.: Conceptual model for spatial data cubes: a UML profile and its automatic implementation. Comput. Stand. Interfaces **38**, 113–132 (2015)
5. Cai, H., Gu, Y., Vasilakos, A.V., Xu, B., Zhou, J.: Model-driven development patterns for mobile services in cloud of things. IEEE Trans. Cloud Comput. **6**(3), 771–784 (2016)
6. Ciccozzi, F., Spalazzese, R.: MDE4IoT: supporting the internet of things with model-driven engineering. IDC 2016. SCI, vol. 678, pp. 67–76. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-48829-5_7
7. Cuzzocrea, A., Mazón, J.N., Trujillo, J., Zubcoff, J., et al.: Model-driven data mining engineering: from solution-driven implementations to 'composable' conceptual data mining models. Int. J. Data Min. Model. Manag. **3**(3), 217–251 (2011)
8. Golfarelli, M., Rizzi, S., Turricchia, E.: Modern software engineering methodologies meet data warehouse design: 4WD. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 66–79. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23544-3_6
9. Lennerholt, C., van Laere, J., Söderström, E.: Implementation challenges of self service business intelligence: a literature review. In: 51st Hawaii International Conference on System Sciences, Hilton Waikoloa Village, Hawaii, USA, 3–6 January 2018, vol. 51, pp. 5055–5063. IEEE Computer Society (2018)
10. Marouane, H., Makni, A., Bouaziz, R., Duvallet, C., Sadeg, B.: Definition of design patterns for advanced driver assistance systems. In: Proceedings of the 10th Travelling Conference on Pattern Languages of Programs, p. 3. ACM (2016)
11. Mezghani, E., Exposito, E., Drira, K.: A model-driven methodology for the design of autonomic and cognitive iot-based systems: Application to healthcare. IEEE Trans. Emerg. Top. Comput. Intell. **1**(3), 224–234 (2017)
12. Nguyen, X.T., Tran, H.T., Baraki, H., Geihs, K.: Frasad: a framework for model-driven IoT application development. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), pp. 387–392. IEEE (2015)
13. Patel, P., Cassou, D.: Enabling high-level application development for the internet of things. J. Syst. Softw. **103**, 62–84 (2015)
14. Plazas, J.E., Bimonte, S., De Sousa, G., Corrales, J.C.: Data-centric UML profile for wireless sensors: application to smart farming. Int. J. Agri. Environ. Inf. Syst. (IJAEIS) **10**(2), 21–48 (2019)
15. Saggi, M.K., Jain, S.: A survey towards an integration of big data analytics to big insights for value-creation. Inf. Process. Manag. **54**(5), 758–790 (2018)
16. Taktak, S., Alshomrani, S., Feki, J., Zurfluh, G.: The power of a model-driven approach to handle evolving data warehouse requirements. In: 5th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2017), pp. 169–181. SciTePress (2017)
17. Thramboulidis, K., Christoulakis, F.: Uml4iot–a UML-based approach to exploit iot in cyber-physical manufacturing systems. Comput. Ind. **82**, 259–272 (2016)

Semantic Web

Consistency and Certain Answers in Relational to RDF Data Exchange with Shape Constraints

Iovka Boneva, Sławek Staworko, and Jose Lozano(✉)

Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, 59000 Lille, France
jose-martin.lozano-aparicio@univ-lille.fr

Abstract. We investigate the data exchange from relational databases to RDF graphs inspired by R2RML with the addition of target shape schemas capturing fragments of SHACL and ShEx. We study the problems of *consistency* i.e., checking that every source instance admits a solution, and *certain query answering* i.e., finding answers present in every solution. We identify the class of *constructive relational to RDF data exchange* that uses IRI constructors and full tgds (with no existential variables) in its source to target dependencies. We show that the consistency problem is coNP-complete. We introduce the notion of *universal simulation solution* that allows to compute certain query answers to any class of queries that is *robust under simulation*. One such class are nested regular expressions (NREs) that are *forward* i.e., do not use the inverse operation. Using universal simulation solution renders tractable the computation of certain answers to forward NREs (data-complexity).

1 Introduction

The recent decade has seen RDF raise to the task of interchanging data between Web applications [25]. In many applications the data is stored in a relational database and only exported as RDF, as evidenced by the proliferation of languages for mapping relational databases to RDF, such as R2RML [18], Direct Mapping [4] or YARRRML [20]. As an example, consider the following R2RML mapping, itself in RDF format presented in Turtle syntax

```
<#EmpMap>
  rr:logicalTable [ rr:sqlQuery "SELECT id, name, email FROM Emp NATURAL JOIN Email" ];
  rr:subjectMap [ rr:template "emp:{id}"; rdf:type :TEmp ];
  rr:predicateObjectMap [ rr:predicate :name; rr:objectMap [ rr:column "name"] ];
  rr:predicateObjectMap [ rr:predicate :email; rr:objectMap [ rr:column "email"] ].
```

It exports the join of two relations $Emp(\underline{id}, name)$ and $Email(\underline{id}, name)$ into a set of triples. For every employee it creates a dedicated Internationalized Resource

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 97–107, 2020.
https://doi.org/10.1007/978-3-030-54623-6_9

Identifier (IRI) consisting of the prefix `emp:` and the employee identifier. More importantly, the class (`rdf:type`) of each employee IRI is declared as :TEmp.

RDF has been originally proposed schema-less to promote its adoption but the need for schema languages for RDF has been since identified and deemed particularly important in the context of exchange of data between applications [23,32].

One family of proposed schema formalisms for RDF is based on *shape constraints* and this class includes *shape constraint language* (SHACL) [17,22] and *shape expressions schemas* (ShEx) [11,27,28]. The two languages allow to define a set of types that impose structural constraints on nodes and their immediate neighborhood in an RDF graph. For instance, the type :TEmp may be defined as

$$\text{:TEmp} \quad \{\ \text{:name xsd:string; :email xsd:string?; :works @:TDept+}\ \}$$

Essentially, an employee IRI must have a single :name property, an optional :email property that are both strings, and at least one :works property each leading to an IRI of type :TDept.

In the present paper we formalize the process of exporting a relational database to RDF as *data exchange*, and study two of its fundamental problems: *consistency* and *certain query answering*. In data exchange the mappings from the source database to the target database are modeled with *source-to-target tuple-generating dependencies* (st-tgds). For mappings defined with R2RML we propose a class of *full constructive* st-tgds, which use *IRI constructors* to map entities from the relational database to IRIs in the RDF. For instance, the R2RML mapping presented before can be expressed with the following st-tgd

$$\begin{aligned} \mathit{Emp}(\mathit{id}, \mathit{name}) \wedge \mathit{Email}(\mathit{id}, \mathit{email}) \Rightarrow\ & \mathit{Triple}(\mathit{emp2iri}(\mathit{id}), \text{:name}, \mathit{name}) \wedge \\ & \mathit{Triple}(\mathit{emp2iri}(\mathit{id}), \text{:email}, \mathit{email}) \wedge \\ & \mathsf{TEmp}(\mathit{emp2iri}(\mathit{id})), \end{aligned}$$

where *emp2iri* is an IRI constructor that generates an IRI for each employee. To isolate the concerns, in our analysis of the st-tgds we refrain form inspecting the definitions of IRI constructors and require only that they are *non-overlapping*, i.e. no two IRI constructors are allowed to output the same IRI. We call the above setting *constructive relational to RDF data exchange*. We report that in this setting all 4 use cases of R2RML [6] can be expressed. Furthermore, we can cover 38 out of 54 test cases for R2RML implementations [31]. Among the non-covered test cases, 9 use pattern-based function to transform data values and 7 use SQL statements with aggregation functions. In fact, our assessment is that the proposed framework allows to fully address all but one out of the 11 core functional requirements for R2RML [6], namely the *Apply a Function before Mapping*. Finally, in our investigation we consider a class of *shape schemas* that are at the intersection of SHACL and ShEx. They are known to have desirable computational properties while remaining practical, and furthermore, they posses a sought-after feature of having an equivalent graphical representation (in the form of shape graphs) [29].

For a given source relational instance, a *solution* to data exchange is a target database (an RDF graph in our case) that satisfies the given set of st-tgds and the target schema (a shape schema in our case). The number of solutions may vary from none to infinitely many. The problem of *consistency* is motivated by the need for static verification tools that aim to identify potentially erroneous data exchange settings that are *inconsistent* i.e., admit no solution for some source database instance. In general, a consistent data exchange setting may yield many solutions to a given source instance and it is commonplace to apply the *possible world semantics* [21] to evaluate queries: a *certain answer* is an answer returned in every solution. It is standard practice to construct a solution that allows to easily compute certain answers. In the case of relational data exchange, *universal solutions* have been identified and allow to easily compute certain answers to conjunctive queries, or any class of queries preserved under homomorphism for that matter [19]. Unfortunately, for relational to RDF data exchange with target shape schema, a finite universal solution might not exists, even if the setting is consistent and admits solutions. Also, the class of conjunctive queries, while adequate for expressing queries for relational databases, is less so for RDF. Query languages, like SPARQL, allowing regular path expressions with nesting have been proposed to better suit the needs of querying RDF [26].

Our contributions are as follows. We formalize the framework of relational to RDF data exchange with target shape schema and IRI constructors, and we identify the class of *constructive relational to RDF data exchange* that uses shape schemas and full constructive source-to-target dependencies. We provide an effective characterization of consistency of constructive relational to RDF data exchange settings and show that the problem is coNP-complete. We propose a novel notion of *universal simulation solution*, that can be constructed for any consistent constructive relational to RDF data exchange setting, and use it to show tractability of computing certain answers to *forward nested regular expressions*. In an extended version of the present paper [14] we present full details and study further a number of extensions of our framework and show each time negative computational consequences.

2 Preliminaries

In this section we recall the standard notions of logic and databases [1,24].

Relational Databases. A *relational schema* is a pair $\mathbf{R} = (\mathscr{R}, \Sigma_{\mathrm{fd}})$ where $\mathscr{R}$ is a set of relation names and Σ_{fd} is a set of functional dependencies. Each relation name has a fixed arity and a set of attribute names. A *functional dependency* is written as usual $R : X \to Y$ where R is a relation name and X and Y are two sets of attributes of R. An *instance* I of $\mathbf{R}$ is a function that maps every relation name of $\mathbf{R}$ to a set of tuples over a set Lit of constants (also called literal values). The instance I is *consistent* if it satisfies all functional dependencies Σ_{fd}. In the sequel, we often view an instance as a relational structure over the signature $\mathscr{R}$.

Graphs. An RDF graph G is a labeled graph whose nodes are divided into two *kinds*: *literal* nodes (Lit) and *non-literal* (Iri) nodes with only the latter allowed to have outgoing edges. Every node is labeled and we adopt the *unique name assumption* (UNA) i.e., no two node have the same label. Consequently, we equate nodes with their labels and by $\mathit{nodes}(G)$ we denote the set of labels of nodes of G. Also, each edge is labeled with a predicate name, which is a non-null resource name Pred. As a result of using chase some nodes may be labeled with names nulls.

Shape Schemas. A *shapes schema* is a pair $\mathbf{S} = (\mathscr{T}, \delta)$, where $\mathscr{T}$ is a finite set of *type names* and $\delta : \mathscr{T} \times \mathsf{Pred} \to (\mathscr{T} \cup \{\mathit{Literal}\}) \times \{\mathtt{1}, \mathtt{?}, \mathtt{*}, \mathtt{+}\}$ defines shape constraints. A *shape constraint* $\delta(T, p) = (S, \mu)$, often presented as $\delta(T, p) = S^{\mu}$, reads as follows: if a node has type T, then every neighbor reached with an outgoing p-edge must have type S and the number of such neighbors must be within the bounds of μ: precisely one if $\mu = \mathtt{1}$, at most one if $\mu = \mathtt{?}$, at least one if $\mu = \mathtt{+}$, and arbitrarily many if $\mu = \mathtt{*}$. Whenever $\mu = \mathtt{1}$ or $\mu = \mathtt{?}$ we say that the predicate p is *functional* for type T.

Dependencies. We employ the standard syntax of first-order logic (cf. [24]).

In the sequel, we shall view graphs as relational structures using the ternary predicate *Triple* and monadic predicates in $\mathscr{T} \cup \{\mathit{Literal}\}$ to indicate the types of nodes. Furthermore, in formulas we use the edge labels Pred as constant symbols. Later on, we additionally introduce functions that allow to map the values in relational databases to resource names used in RDF graphs, and we allow the use of function names in formulas but without nesting.

Now, a *dependency* is a formula of the form $\forall \bar{x}.\varphi \Rightarrow \exists \bar{y}.\psi$, where φ is called the *body* and ψ the *head* of the dependency, and we typically omit the universally quantified variables and write simply $\varphi \Rightarrow \exists \bar{y}.\psi$. A dependency is *equality-generating* (egd) if its body is a clause and its head consists of an equality condition $x = y$ on a pair of variables. A *tuple-generating dependency* (tgd) uses clauses in both its head and its body. A tgd is *full* if it has no existentially quantified variables.

A number of previously introduced concepts can be expressed with dependencies. Any functional dependency is in fact an equality-generating dependency. Interestingly, any deterministic shape schema $\mathbf{S}$ can be expressed with a set $\Sigma_{\mathbf{S}}$ of equality- and tuple-generating dependencies. More precisely, whenever $\delta(T, p) = S^{\mu}$ the set $\Sigma_{\mathbf{S}}$ contains:

(TP) the *type propagation* rule: $T(x) \wedge \mathit{Triple}(x, p, y) \Rightarrow S(y)$,
(PF) the *predicate functionality* rule if $\mu = \mathtt{1}$ or $\mu = \mathtt{?}$:
$T(x) \wedge \mathit{Triple}(x, p, y_1) \wedge \mathit{Triple}(x, p, y_2) \Rightarrow y_1 = y_2$,
(PE) the *predicate existence* rule if $\mu = \mathtt{1}$ or $\mu = \mathtt{+}$: $T(x) \Rightarrow \exists y.\ \mathit{Triple}(x, p, y)$.

3 Constructive Relational to RDF Data Exchange

An n-ary *IRI constructor* is a function $f : \mathsf{Lit}^n \to \mathsf{Iri}$ that maps an n-tuple of database constants to an RDF resource name. A *IRI constructor* library is a pair $\mathbf{F} = (\mathscr{F}, F)$, where $\mathscr{F}$ is a set of IRI constructor names and F is their interpretation. $\mathbf{F}$ is *non-overlapping* if all its IRI constructors have pairwise disjoint ranges.

Definition 1. A *relational to RDF data exchange setting with fixed IRI constructors* is a tuple $\mathscr{E} = (\mathbf{R}, \mathbf{S}, \Sigma_{\mathrm{st}}, \mathbf{F})$, where $\mathbf{R} = (\mathscr{R}, \Sigma_{\mathrm{fd}})$ is a source relational schema, $\mathbf{S} = (\mathscr{T}, \delta)$ is a target shape constraint schema, $\mathbf{F} = (\mathscr{F}, F)$ is an IRI constructor library, and Σ_{st} is a set of *source-to-target tuple generating dependencies (*st-tgds*)* whose bodies are formulas over $\mathscr{R}$ and heads are formulas over $\mathscr{F} \cup \mathscr{T} \cup \{\mathit{Literal}\}$ without nesting of function symbols in $\mathscr{F}$. $\mathscr{E}$ is *constructive* if the library of IRI constructors is non-overlapping and the st-tgds Σ_{st} are full tgds. A typed graph J is a *solution* to $\mathscr{E}$ for a source instance I of $\mathbf{R}$, iff J satisfies $\mathbf{S}$ and $I \cup J \cup F \models \Sigma_{\mathrm{st}}$. By $sol_{\mathscr{E}}(I)$ we denote the set of all solutions for I to $\mathscr{E}$. □

In the reminder we fix a constructive data exchange setting $\mathscr{E}$, and in particular, we assume a fixed library of IRI constructors $\mathbf{F}$. Since we work only with constructive data exchange settings, w.l.o.g. we can assume that the heads of all st-tgds consist of one atom only. We point out that while a constructive data exchange setting does not use egds, our constructions need to accommodate egds and tgds coming from the shapes schema.

The standard *chase* procedure allows to construct a solution to $\mathscr{E}$ for a source instance I. However, such solution might not exist either because the chase fails due to an unsatisfiable egd, or because it never terminates. The *core pre-solution* for I to $\mathscr{E}$ is the result J_0 of chase on I with the st-tgds Σ_{st} and all **TP** rules of $\mathbf{S}$. In essence J_0 is obtained by exporting the relational data to RDF triples with Σ_{st} and then propagating any missing types according to $\mathbf{S}$ but without creating any new nodes with **PE** rules. This process does not introduce any null values and always terminates yielding a unique result. Naturally, J_0 is included in any solution $J \in sol_{\mathscr{E}}(I)$.

4 Consistency

Recall that a data exchange setting $\mathscr{E}$ is *consistent* if every consistent source instance I of $\mathbf{R}$ admits a solution to $\mathscr{E}$.[1] Throughout this section we fix a data exchange setting $\mathscr{E} = (\mathbf{R}, \mathbf{S}, \Sigma_{\mathrm{st}}, \mathbf{F})$ and study its consistency.

Note that only predicate functionality egds can bring inconsistency. We show that $\mathscr{E}$ is inconsistent if and only if there is a source instance I and a predicate functionality egd $T(x) \wedge \mathit{Triple}(x, p, y) \wedge \mathit{Triple}(x, p, y') \Rightarrow y = y'$ in $\mathscr{E}$ such that all solutions to $\mathscr{E}$ for I need to satisfy the body of the above rule but y and y' are not equatable. We distinguish two situations, that together provide a necessary and sufficient condition for $\mathscr{E}$ to be (in-)consistent.

[1] This was called absolute consistency in [9].

- *value inconsistency* when y and y' are different constants (see Sect. 4),
- *node kind inconsistency* when one among y, y' is a literal and the other is not (see Sect. 4).

Value Consistency. Let $\Sigma_{\mathbf{S}}^{\mathsf{PF}}$ be the set of predicate functionality rules from $\Sigma_{\mathbf{S}}$ as defined in Sect. 3.

An instance I of $\mathbf{R}$ is called *value inconsistent* if $J \not\models \Sigma_{\mathbf{S}}^{\mathsf{PF}}$, where J is the core pre-solution for I to $\mathscr{E}$. The data exchange setting $\mathscr{E}$ is called *value inconsistent* if there exists a consistent instance I of $\mathbf{R}$ that is value inconsistent.

We now sketch a decision procedure *value-inconsistent*$(\mathscr{E})$ that tests whether $\mathscr{E}$ is value inconsistent by constructing a value inconsistent source instance whenever such exists. It is illustrated on the following example data exchange setting. The relational signature contains symbols R, S both of arity two, and the IRI constructors are $\{g_0, g, f\}$ all of arity one. The shapes schema $\mathbf{S}$ is given by $\delta(U_0, r) = U^*$, $\delta(U, q) = T^*$, $\delta(T, p) = \mathit{Literal}^1$, and the st-tgds are:

$$\begin{array}{ll}
(\sigma_1)\ R(x_0, x_1) \Rightarrow U_0(g_0(x_1)) & R(x,z) \land S(x,y') \Rightarrow \mathit{Triple}(f(x), p, y')\ (\sigma) \\
(\sigma_2)\ R(x_1, x_2) \Rightarrow \mathit{Triple}(g_0(x_1), r, g(x_2)) & S(x,y) \Rightarrow \mathit{Triple}(f(x), p, y)\ (\sigma') \\
(\sigma_3)\ R(x_2, x) \Rightarrow \mathit{Triple}(g(x_2), q, f(x)) &
\end{array}$$

Using that IRI constructors have pairwise disjoint ranges, it follows from the definition that a source instance is value inconsistent iff its core pre-solution contains a triple of facts of the form $\{T(f(x)), \mathit{Triple}(f(x), p, b), \mathit{Triple}(f(x), p, b')\}$ for some function symbol f, and such that x and $y \neq y'$ are constants and the predicate p is functional for type T in $\mathbf{S}$. We call such triple of facts a *violation* and (T, p, f) its *sort*. There is a finite number of possible violation sorts for $\mathscr{E}$. On the example, the possible violation sorts are $(T, p, f), (T, p, g_0)$ and (T, p, g), as the unique functional predicate rule is for type T and predicate p. The procedure *value-inconsistent*$(\mathscr{E})$ enumerates all violation sorts and for each of them tries to build a value inconsistent instance.

So consider the violation sort (T, p, f). Although none of the st-tgds heads contains a fact of the form $T(f(\_))$, we remark that such fact can be obtained using type propagation rules. Indeed, consider the source instance $I' = \{R(x_0, x_1), R(x_1, x_2), R(x_2, x)\}$ obtained as the union of the bodies of st-tgds (σ_1), (σ_2) and (σ_3), where variables are used as constants. Applying on I' the st-tgds (σ_1), (σ_2) and (σ_3) together with the type propagation rules for U_0, r and U, q yields the target instance $J' = I' \cup \{U_0(g_0(x_1)), \mathit{Triple}(g_0(x_1), r, g(x_2)), U(g(x_2)), \mathit{Triple}(g(x_2), q, f(x)), T(f(x))\}$ which does contain a fact $T(f(x))$ as required. We show that for given T and f, a fact of the form $T(f(\_))$ exists in some core pre-solution w.r.t. $\mathscr{E}$ if and only if $\mathscr{E}$ contains an appropriate finite and elementary (i.e. without repetitions) sequence of st-tgds such as (σ_1), (σ_2), (σ_3) above that, combined with type propagation rules, allows to obtain $T(f(\_))$.

Now we need to add facts of the forms $\mathit{Triple}(f(x), p, y)$ and $\mathit{Triple}(f(x), p, y')$. This is done using the st-tgds (σ) and (σ'), called *contentious* for f and p. So consider $I = I' \cup \{S(x, y), R(x, z), S(x, y')\}$ obtained by adding the bodies of (σ) and (σ') to I'. Its core pre-solution is

$J = J' \cup \{\mathit{Triple}(f(x), p, y), \mathit{Triple}(f(x'), p, y'), \ \mathit{Literal}(y), \mathit{Literal}(y')\}$ and it contains a violation of sort (T, p, f) whenever $y \neq y'$.

Thus, I is an instance over the source signature that is value inconsistent. This does not imply yet that $\mathscr{E}$ is inconsistent, as I might not be consistent w.r.t the source functional dependencies. If the first attribute of S is a primary key, then $y = y'$, I is not value inconsistent and we can actually show that the example data exchange setting $\mathscr{E}$ is consistent. Otherwise, I is value inconsistent, and so is $\mathscr{E}$.

Here is a NP procedure that checks value inconsistency by guessing an inconsistent source instance:

- guess a violation sort (T, p, f) of $\mathscr{E}$,
- guess an elementary sequence of st-tgds $\sigma_1, \ldots, \sigma_n$ that allows to generate a fact $T(f(\_))$,
- guess two st-tgds σ, σ' contentious for f and p,
- construct in PTIME a source instance I as the union of the bodies of $\sigma_1, \ldots, \sigma_n, \sigma, \sigma'$ (after appropriate renaming of variables),
- show in PTIME that I satisfies the source functional dependencies.

Theorem 1. *Value consistency of a data exchange setting is in coNP.*

Node Kind Consistency. Node kind inconsistency is specific to relational to RDF data exchange due to the presence of two types of values, namely IRIs and literals. It corresponds to situations in which two values are equated by a predicate functionality rule while one of them is a literal (that is, has type *Literal*) but the other is not (that is, has a type $T \in \mathscr{T}$). Therefore we identify node kind inconsistency with the fact that a node has both types *Literal* and $T \neq \mathit{Literal}$.

For a typed graph J and a node $n \in \mathit{nodes}(J)$, let $\mathit{types}_J(n) = \{T \in \mathscr{T} \cup \{\mathit{Literal}\} \mid T(n) \in J\}$. For a source instance I, define its sets of *co-occurring types* as $\mathit{CoTypes}(I) = \{\mathit{types}_J(n) \mid J \in \mathit{sol}_{\mathscr{E}}(I),\ n \in \mathit{nodes}(J)\}$. Let $\mathit{CoTypes}(\mathscr{E}) = \bigcup_{\text{I instance of } \mathbf{R}} \mathit{CoTypes}(I)$.

A source instance I is called *node kind inconsistent* if $\mathit{CoTypes}(I)$ contains a set X s.t. $\{\mathit{Literal}, T\} \subseteq X$ for some T in $\mathscr{T}$. The data exchange setting $\mathscr{E}$ is called node kind inconsistent if there is a node kind inconsistent instance I of $\mathbf{R}$.

We show that $\mathscr{E}$ is node kind inconsistent iff $\mathit{CoTypes}(\mathscr{E})$ contains a set X such that $\{\mathit{Literal}, T\} \subseteq X$ for some T in $\mathscr{T}$. Furthermore, for any $X \subseteq \mathscr{T} \cup \{\mathit{Literal}\}$, we can test in PTIME whether X belongs to $\mathit{CoTypes}(\mathscr{E})$. Therefore,

Proposition 1. *Checking node kind inconsistency of $\mathscr{E}$ is in NP.*

The central claim of this section is Theorems 2 below. The lower bound is shown using a reduction to the complement of SAT.

Theorem 2. *A constructive relational to RDF data exchange setting $\mathscr{E}$ is consistent iff it is value consistent and node kind consistent. Checking consistency of a constructive relational to data exchange setting is decidable and coNP-complete.*

5 Certain Query Answering

In this section we investigate computing certain answers to Boolean queries focusing on a subclass of nested regular expressions (NREs). In [14] we show that our results extend to non-Boolean queries but also that handling the full class of NREs leads to an increase in computational complexity.

Throughout this section we fix a constructive data exchange setting $\mathscr{E} = (\mathbf{R}, \mathbf{S}, \Sigma_{\mathrm{st}}, \mathbf{F})$ and assume $\mathscr{E}$ is consistent. We recall that for a Boolean graph query Q, *true* is the *certain answer* to Q in I w.r.t. $\mathscr{E}$ iff *true* is the answer to Q in every solution to $\mathscr{E}$ for I.

The standard approach to computing certain answers is to construct a universal solution with the chase and evaluate the query against it (and to drop any answers with null values) [19]. However, in our case a finite universal solution may not exist because the chase may enter an infinite loop due to **PE** rules when the shape schema is strongly-recursive i.e., it has a cycle with multiplicities of `1` and `+`. Infinite chase corresponds to an attempt to unravel such cycles by inventing new nodes ad infinitum. Instead, we construct a solution where a new node is invented only if one satisfying precisely the same types has not been invented before. Such a solution is not universal, but interestingly, it has a different flavor of universality, one that can be captured with the standard notion of graph simulation: any solution can be simulated in it. We also show that this notion of universality is good enough for classes of queries that are robust under simulation, and we identify a practical class of forward nested regular expressions with this property. This yields a practical class of queries with tractable certain answers.

Nested Regular Expressions. In this paper we work with the class of *nested regular expressions* (NREs) that have been proposed as the navigational core of SPARQL [26]. In essence, NREs are regular expressions that use concatenation $\cdot$, union $+$, Kleene's closure $*$, inverse $-$, and permit nesting and testing node and edge labels. We refer the reader to [14] for detailed definition.

We point out that NREs are incompatible with conjunctive queries but even forward NREs capture the subclass of acyclic conjunctive queries. Also, forward NREs properly capture regular path queries.

Graph Simulation and Robust Query Classes. We adapt the classic notion of graph simulation to account for null values. Formally, a *simulation* of a graph G by a graph H is a relation $R \subseteq nodes(G) \times nodes(H)$ such that for any $(n, m) \in R$, we have 1) n is a literal node if and only if m is a literal node, 2) if n is not null, then m is not null and $n = m$, and 3) for any outgoing edge from n with label p that leads to n' there is a corresponding outgoing edge from m with label p that leads to m' such that $(n', m') \in R$. The set of simulations is closed under union, and consequently, there is always one maximal simulation, and if (n, m) is contained in it, we say that n is simulated by m.

Also, we say that G is *simulated* by H if every node of G is simulated by a node of H. We are interested in simulations because they capture the essence of exploring a graph by means of following outgoing edges only.

Definition 2. A class $\mathscr{Q}$ of Boolean queries on graphs is *robust under simulation* iff for any query $Q \in \mathscr{Q}$ and any two graph G and H such that G is simulated by H, if Q is true in G, then Q is true in H. □

Naturally, the class of forward NREs has this very property.

Lemma 1. *The class of forward nested regular expressions is robust under simulation.*

Universal Simulation Solution. When dealing with classes of queries that are robust under simulation we employ simulation instead of homomorphism to define a solution that allows to find all certain answers.

Definition 3. A typed graph $\mathscr{U}$ is a *universal simulation solution* to $\mathscr{E}$ for I iff $\mathscr{U}$ is simulated by every solution J to $\mathscr{E}$ for I. □

And indeed, a universal simulation solution does allow us to capture certain answers for queries from classes robust under simulation.

Theorem 3. *Let $\mathscr{Q}$ be a class of Boolean graph queries robust under simulation. For any query $Q \in \mathscr{Q}$ and any consistent instance I of* $\mathbf{R}$*, true is the certain answer to Q in I w.r.t. $\mathscr{E}$ if and only if true is the answer to Q in a universal simulation solution to $\mathscr{E}$ for I.*

The main challenge is in constructing a universal simulation solution. The precise construction is presented in [14] and we outline it roughly. First we begin with the core pre-solution that is obtained from the source instance I with the the st-tgds Σ_{st} and the **TP** rules for **S** that propagate the types according to the shape schema. Then, we add fresh null values that ensure satisfaction of the schema, as required by the **PE** rules for **S**. We point out that each null node corresponds to a subset of types of **S** that it needs to satisfy, which bounds their number by $2^{|\mathbf{S}|}$, and furthermore, using the Chinese reminder theorem we show that this bound is tight. To ensure that the produced universal simulation solution has the smallest size, we employ the standard technique of quotient by bisimulation of the obtained graph [30].

Theorem 4. *For an instance I of* $\mathbf{R}$*, we can construct a size-minimal universal simulation solution $\mathscr{U}_0$ in time polynomial in the size of I and exponential in the size of* $\mathbf{S}$*. The size of $\mathscr{U}$ is bounded by a polynomial in the size of I and an exponential function in the size of* $\mathbf{S}$*.*

Complexity. We can now characterize the data complexity of certain query answering. Recall that data complexity assumes the query and the data exchange setting to be fixed, and thus of fixed size, and only the source instance is given on the input. Consequently, the size of universal simulation solution $\mathscr{U}_0$ is polynomially-bounded by the size of I. Since the data complexity of evaluating NREs is know to be PTIME [26], we get the following result.

Theorem 5. *The data complexity of computing certain answers to forward nested regular expressions w.r.t. constructive relational to RDF data exchange setting is in PTIME.*

6 Related Work and Conclusions

R2RML is a W3C standard language for defining custom relational to RDF mappings [18], other languages such as YARRRML [20] are compiled to R2RML but they do not consider target constraints. Data exchange has been considered for graph databases with varying expressive power of mapping formalisms such as NREs [7,10], which is however incomparable with shape schemas. In [12] we have considered consistency for fully-typed constructive data exchange settings. In [13] we have demonstrated a graphical tool for defining constructive relational to RDF mappings. Our results do not follow from the exists results on the standard relational data exchange [19], which are either too limited in their expressive power [15,19] or come with a significant complexity penalty [2,3,5,8,9,16].

We have presented a data exchange framework for modeling R2RML scripts, we have studied the problems of consistency and certain query answering, and characterized their complexity. In [14] we also show that extending the framework in a number of natural directions generally leads to an increase of complexity.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Boston (1995)
2. Amano, S., David, C., Libkin, L., Murlak, F.: XML schema mappings: data exchange and metadata management. J. ACM **61**(2), 12:1–12:48 (2014)
3. Arenas, M., Barceló, P., Reutter, J.L.: Query languages for data exchange: beyond unions of conjunctive queries. Theory Comput. Syst. **49**(2), 489–564 (2011)
4. Arenas, M., Bertails, A., Prud'hommeaux, E., Sequeda, J.: A direct mapping of relational data to RDF. W3C Recomm. **27**, 1–11 (2012)
5. Arenas, M., Libkin, L.: XML data exchange: consistency and query answering. J. ACM **55**(2), 7:1–7:72 (2008)
6. Auer, S., Feigenbaum, L., Miranker, D., Fogarolli, A., Sequeda, J.: Use cases and requirements for mapping relational databases to RDF, W3C (2010)
7. Barceló, P., Pérez, J., Reutter, J.L.: Schema mappings and data exchange for graph databases. In: International Conference on Database Theory (ICDT), pp. 189–200 (2013)
8. Bienvenu, M., Ortiz, M., Simkus, M.: Regular path queries in lightweight description logics: complexity and algorithms. J. Artif. Intell. Res. **53**, 315–374 (2015)
9. Bojańczyk, M., Kołodziejczyk, L.A., Murlak, F.: Solutions in XML data exchange. J. Comput. Syst. Sci. **79**(6), 785–815 (2013)
10. Boneva, I., Bonifati, A., Ciucanu, R.: Graph data exchange with target constraints. In: EDBT/ICDT Workshops (GraphQ), pp. 171–176 (2015)
11. Boneva, I., Labra Gayo, J.E., Prud'hommeaux, E.G.: Semantics and validation of shapes schemas for RDF. In: International Semantic Web Conference (ISWC), pp. 104–120 (2017)
12. Boneva, I., Lozano, J., Staworko, S.: Relational to RDF data exchange in presence of a shape expression schema. In: Alberto Mendelzon International Workshop (AMW) (2018)
13. Boneva, I., Lozano, J., Staworko, S.: ShERML: mapping relational data to RDF. In: ISWC Satellite Tracks, pp. 213–216 (2019)

14. Boneva, I., Staworko, S., Lozano, J.: Consistency and certain answers in relational to RDF data exchange with shape constraints. Technical report. arXiv:2003.13831, April 2020
15. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. J. Web Semant. **14**, 57–83 (2012)
16. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics via alternating tree-automata. Inf. Comput. **237**, 12–55 (2014)
17. Corman, J., Reutter, J.L., Savkovic, O.: Semantics and validation of recursive SHACL. In: International Semantic Web Conference, pp. 318–336 (2018)
18. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C Recomm. (2011). https://www.w3.org/TR/r2rml/
19. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theoret. Comput. Sci. **336**, 89–124 (2005)
20. Heyvaert, P., De Meester, B., Dimou, A., Verborgh, R.: Declarative rules for Linked Data generation at your fingertips! In: The Semantic Web: ESWC Satellite Events, pp. 213–217, June 2018
21. Imieliński, T., Lipski Jr., W.: Incomplete information in relational databases. J. ACM **31**(4), 761–791 (1984)
22. Knublauch, H., Kontokostas, D.: Shapes constraint language (SHACL). W3C Recomm. (2017). https://www.w3.org/TR/shacl/
23. Labra Gayo, J.E., Prud'hommeaux, E., Boneva, I., Kontokostas, D.: Validating RDF Data. Synthesis Lectures on the Semantic Web: Theory and Technology (2017)
24. Libkin, L.: Elements of Finite Model Theory. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-662-07003-1
25. Michel, F., Montagnat, J., Faron Zucker, C.: A survey of RDB to RDF translation approaches and tools. Technical report, University Sophia Antipolis (2013)
26. Pérez, J., Arenas, M., Gutierrez, C.: nSPARQL: a navigational language for RDF. J. Web Semant. **8**(4), 255–270 (2010)
27. Prud'hommeaux, E., Boneva, I., Emilio, J.L.G., Kellogg, G.: Shape expressions language 2.1. W3C Draft (2018)
28. Staworko, S., Boneva, I., Labra Gayo, J.E., Hym, S., Prud'hommeaux, E.G., Solbrig, H.R.: Complexity and expressiveness of ShEx for RDF. In: International Conference on Database Theory (ICDT), pp. 195–211 (2015)
29. Staworko, S., Wieczorek, P.: Containment of shape expression schemas for RDF. In: ACM Symposium on Principles of Database Systems (PODS), pp. 303–319 (2019)
30. Tzitzikas, Y., Lantzaki, C., Zeginis, D.: Blank node matching and RDF/S comparison functions. In: International Semantic Web Conference (ISWC), pp. 591–607 (2012)
31. Villazón, B., Hausenblas, M.: R2RML and direct mapping test cases. W3C (2012)
32. W3C: RDF validation workshop report: practical assurances for quality RDF data (2013)

OWL-T for a Semantic Description of IoT

Zakaria Maamar[1](✉), Noura Faci[2], Ejub Kajan[3], Muhammad Asim[4], and Ayesha Qamar[4]

[1] Zayed University, Dubai, UAE
zakaria.maamar@zu.ac.ae
[2] Université Claude Bernard, Lyon, France
[3] State University of Novi Pazar, Novi Pazar, Serbia
[4] National University of Computer and Emerging Sciences, Islamabad, Pakistan

Abstract. This paper discusses the steps for developing a semantic description of things in the context of Internet-of-Things (IoT). This description is deemed necessary to discover things prior to confirming their participation in complex business scenarios. Existing approaches provide a restrictive view of how things should be semantically described overlooking the fact that things operate in a complex IoT ecosystem in which they have capabilities to offer and need resources to consume and peers to interact with. To address this restrictive view, Ontology Web Language for Things (OWL-T) is put forward in this paper. OWL-T revolves around 3 dimensions known as interaction, operation, and consumption. Each dimension encompasses abstract conceptual areas that are instantiated using concrete areas, which allows to produce a dedicated IoT ontology.

Keywords: Internet of Things · Semantics · Ontology · OWL

1 Introduction

Internet-of-Things (IoT), the branch of ubiquitous computing that helps people identify and locate things in real-time, real-world environments, has rapidly become the backbone of many smart initiatives that promote better governments, industries, cities, to cite just a few. Gartner mentions that 6.4 billion connected things were in use in 2016, up 3% from 2015, and will reach 20.8 billion by 2020 (www.gartner.com/newsroom/id/3165317). This ever-growing number of things that will, for sure, surpass the number of people on earth, are unfortunately running into the same concerns that other technologies like Web services, had to deal with in the past. Concerns include description, discovery, composition, cognition, vetting, and many others. In this work we examine the semantic description of things in preparation for their discovery and then, composition into complex business scenarios. We resort to the well-defined standard Ontology Web Language for Services (OWL-S) to develop our Ontology Web Language for Things (OWL-T).

Contrarily to OWL-S that semantically describes a service from 3 conceptual areas known as *profile*, *process model*, and *grounding*, OWL-T adopts a semantic description of a thing from 3 dimensions that we refer to as *interaction*, *consumption*, and *operation*. Then, OWL-T refines each dimension into conceptual areas allowing to answer 5 questions related to thing discovery: with whom does a thing interact, what resources does a thing consume, what does a thing do, how does a thing work, and how is a thing invoked? This way of categorizing OWL-T into dimensions and then, areas allows to capture IoT's intrinsic features such as reduced size, restricted connectivity, continuous mobility, limited energy, and constrained storage. Akshay Uttama Nambi et al. talk about similar features that characterize resource-constrained devices that may be unreachable due to intermittent connectivity, mobility, and/or energy constraints [2]. The rest of this paper is organized as follows. Section 2 is an overview of OWL-S and some existing works on semantic IoT. Section 3 defines OWL-T foundations. Finally, Sect. 4 concludes the paper.

2 Background

This section provides an overview of OWL-S basics and then, presents some works on semantic IoT along with highlighting some pending concerns.

2.1 OWL-S in Brief

Martin et al. discuss how OWL-S that aims at injecting semantics into service computing and thus, allowing for instance to address the limitations of Web services syntax discovery [8]. Thanks to OWL-S, a Web service description is unambiguous making it machine understandable in many contexts like enterprise application integration. On top of Web services discovery, OWL-S supports other tasks like invocation, composition, and execution monitoring.

OWL-S's building blocks are 3 capturing what a service does, how a service works, and how to access a service. *Service profile* describes a service in terms of provider, functional properties (i.e., inputs/outputs and pre/post conditions), and non-functional properties (e.g., provided QoS and location). *Service model* describes what happens when the service is carried out. Finally, *Service grounding* provides details about how to interact with the service, via messages, like communication protocol, message formats, and port numbers for binding.

2.2 Related Work on Semantic IoT

In [1], Agarwal et al. propose an ontology to address the lack of semantic interoperability among IoT heterogeneous testbeds. The ontology leverages a number of core concepts from some existing ontologies and taxonomies like Semantic Sensor Network (SSN), M3-lite, and IoT-lite. The core concepts of Agarwal et al.'s ontology are: physical entity, resource, virtual entity, and IoT service. While we agree on the role of some of these concepts like resource in defining an IoT ontology, we are more concerned with describing things semantically than achieving the interoperability of IoT testbeds for joint operations.

In [9], Seydoux et al. consider semantic interoperability as another challenge for IoT that we add to our own list of challenges in [6] like diversity and multiplicity of things' development and communication technologies and limited IoT-platform interoperability. The authors propose IoT-O (O for ontology) to describe devices and their relations and how these devices are strongly bound to the cyber-physical surroundings. Key concepts in IoT-O are, but not limited to, device, software agent, sensor, actuator, service, energy, and lifecycle. While we agree with some IoT-O concepts, we treat sensor and actuator as duties and hence, properties of things and not as things. Moreover, reasoning over thing description for semantic interoperability requires different details than for service discovery and composition.

In [4], Li and Jiang consider that IoT services are different from traditional services because IoT services are directly related to the cyber-physical world. The authors present a context-based approach for IoT service composition and use OWL to develop an IoT context ontology. Context refers to end-user-end QoS properties, computational environment including devices, networks, operating systems, and physical environment (location, time). By satisfying particular contextual constraints, composition meets users' needs expressed as predefined services to run. Particularly, Li and Jiang's work divides service selection into computational context to select the appropriate services and QoS to select the best services according to users' needs. Compared to our work, the authors assume that services are already predefined and, thus, narrow down the service composition problem to selecting services under some contextual constraints.

In [11], Wang et al. develop an ontology to represent knowledge in IoT. They argue that semantic modeling is critical due to the distributed and heterogeneous nature of things exposed as a set of services in compliance with service-oriented computing principles. Their ontology contains 7 building blocks, namely, IoT services, service test, Quality-of-Service (QoS) and Quality of Information (QoI), deployment, system and platform, observation and measurement, IoT resources, and entity of interest and physical locations. Compared to Wang et al.'s work, we do not expose things as services nor adopt QoS. We associate things with duties having each a set of non-functional properties that constitute the Quality-of-Thing (QoT) model of a thing [10]. However, we agree with Wang et al. on the importance of considering resources during IoT semantic modeling.

In [3], Cassar et al. use service-oriented architecture to enable access to IoT-compliant smart devices. The authors note that existing techniques for service discovery are not appropriate for IoT resources due to their limited computation capabilities and operation in dynamic and constrained physical environments. Thus a semantic IoT service representation model needs to be lightweight. Cassar et al. consider sensor, actuator, and other mobile devices as resources that they wrap into Web services referred to as IoT services. We proceed differently by first, separating devices from resources and second, considering sensing and actuating as duties and not things.

3 OWL-T's Three Dimensions

This section provides an overview of OWL-T language and then, details the 3 dimensions that OWL-T encompasses.

3.1 Overview

In Fig. 1, we suggest a high-level representation of OWL-T that revolves around 3 dimensions that are *interaction*, *consumption*, and *operation*. Each dimension includes one to many conceptual areas depending on the expected use of this dimension in the context of semantic description of things. Finally, each conceptual area is instantiated using concrete areas that become effective during the completion of specific thing-related operations like discovery and composition. In the subsequent sections, we detail each dimension in terms of rationale, specification, conceptual areas, properties, etc.

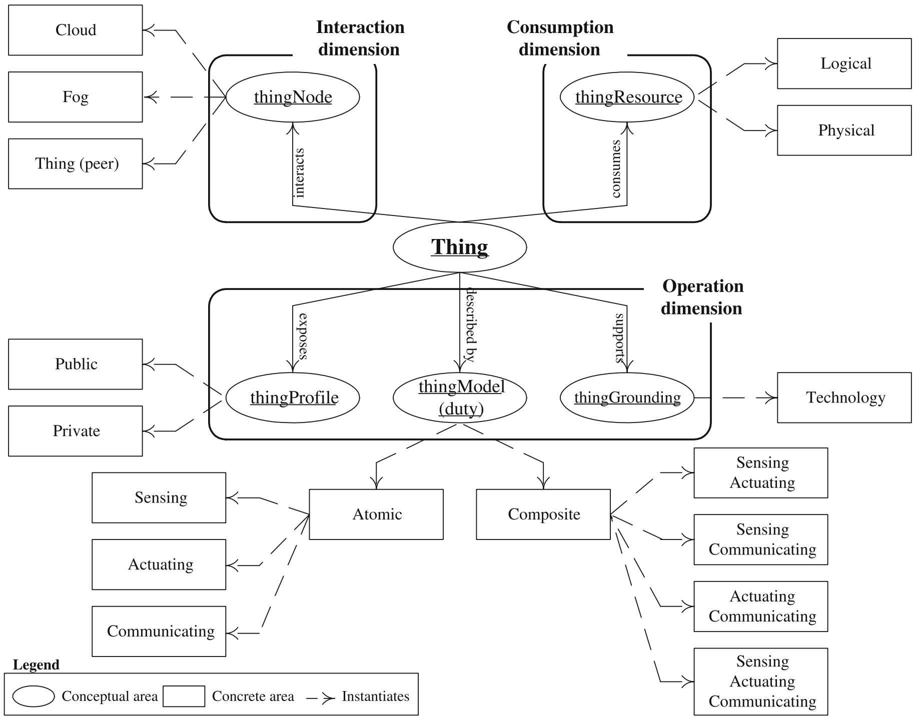


Fig. 1. OWL-T high-level representation

3.2 Interaction Dimension

The rationale of the interaction dimension is to shed light on the stakeholders that form a thing's ecosystem and hence, will engage in interactions with the thing. To establish these stakeholders, the interaction dimension includes one conceptual area referred to as thingNode. It is refined into 3 concrete areas that are *cloud*, *fog*, and *peer* meaning that a thing could interact with cloud platforms (c), fog platforms (f), and other things (t, i.e., peers).

Figure 2 is the OWL-T representation of the interaction dimension featuring one superclass, node, that would encompass any "element" (or stakeholder) residing in a thing's ecosystem. In this figure, the sub-class thingNode to superclass node captures the different types of nodes that a thing could interact with. These nodes are represented with 3 sub-classes referred to as peer, fog, and cloud. Each thingNode has 3 data type properties that are nodeName, textDescription, and nodeCloseness. The last property permits to differentiate if the interaction with a thing is either direct (nodeCloseness is equal to 1) or indirect (nodeCloseness is greater to 1).

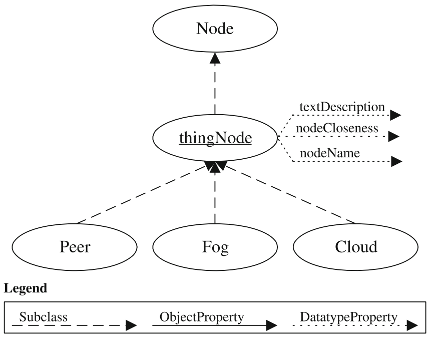


Fig. 2. OWL-T interaction representation

3.3 Operation Dimension

The rationale of the operation dimension is to shed light on the capabilities of things in terms of what they do, how they do what they do, and how they are deployed. To capture these capabilities, the operation dimension includes 3 conceptual areas referred to as thingProfile, thingModel, and thingGrounding. thingProfile is associated with 2 concrete areas that are *public* and *private* meaning that a thing controls its level of exposure to the ecosystem's stakeholders. thingModel is associated with 2 concrete areas that are *atomic* and *composite* meaning that a thing executes operations that are captured as either atomic duties or composite duties to offer to these stakeholders. Finally, thingGrounding is associated with 1 concrete area that is *technology* meaning that a thing runs according to a specific technical specification (hardware and software).

thingModel. In a previous work [5], we captured things' operations with 3 atomic duties that are *sensing* (s), *actuating* (a), and *communicating* (c). These duties are either enabled or disabled depending on IoT applications' requirements ((0,1) in Fig. 3). Simply put, a thing senses the cyber-physical surrounding so that it generates data; a thing actuates data including those that are sensed; and a thing communicates with the cyber-physical surrounding the data that are sensed and/or actuated.

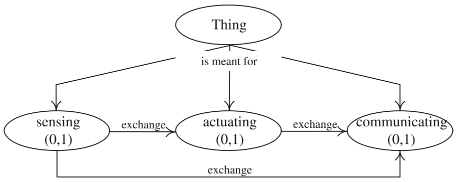


Fig. 3. Representation of a thing's atomic duties

A thing's atomic duties can be put together allowing to form composite duties as per these illustrative cases: *sac* (sensed data are passed on to actuating; and the data that result from actuating are passed on to communicating for sharing); *sa* (sensed data are passed on to actuating; and the data that result from actuating are finals); *sc* (sensed data are passed on to communicating for sharing); and, *ac* (data that result from actuating are passed on to communicating for sharing).

Fig. 4 is the OWL-T representation of the operation dimension with focus on thingModel conceptual area. This representation refers to one super-class, model, that would encompass any "lement" supporting the definition of a thing's model. In this representation as well, the sub-class thingModel to super-class model captures all operations that a thing is expected to perform. These operations correspond to duty class that is specialized into 2 sub-classes referred to as atomic and composite. Both are also specialized into other sub-classes along an object property, chronology, that is associated with the sub-class composite.

thingProfile. In a previous work [10], we developed a Quality-of-Things (QoT (by analogy to Quality-of-Service (QoS)) model to capture a thing's non-functional properties. QoT parameters for sensing include, but not limited to:

- *Frequency of sensing* (e.g., continuous *versus* intermittent).
- *Quality of sensed outcome* that determines for instance, the accuracy and validity of the outcome (e.g., high *versus* low accuracy; high-accuracy outcome would not require any further verification).

QoT properties for actuating include, but not limited to:

- *Quality of actuated outcome* that determines for instance, the accuracy and validity of the outcome.

– *Resource* (e.g., energy, CPU, and storage) consumption during actuating (e.g., high *versus* low energy).

Finally, QoT properties for communicating include, but not limited to:

– *Reception rate of sensed and/or actuated outcome* (incoming flow) that determines for instance, data loss, data volume with respect to a bandwidth, etc.
– *Delivery rate of sensed and/or actuated outcome* (outgoing flow) that determines data loss, data volume with respect to a bandwidth, etc.
– *Resource* (e.g., energy and bandwidth) consumption during communicating (e.g., high *versus* low bandwidth).

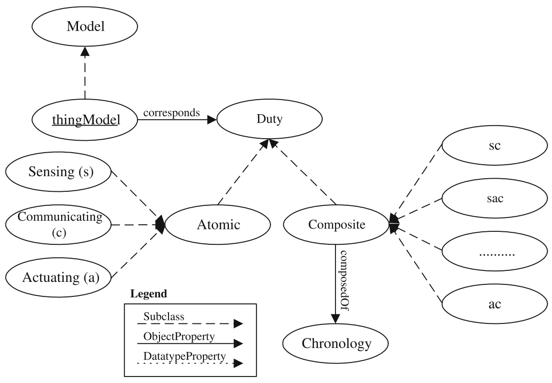


Fig. 4. OWL-T model representation

Fig. 5 is the OWL-T representation of the operation dimension with focus on thingProfile conceptual area. This representation refers to one super-class, profile, that would encompass any "element" supporting the definition of a thing's

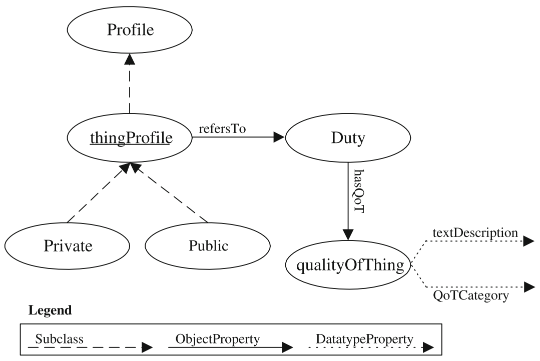


Fig. 5. OWL-T profile representation

profile. In this representation as well, the sub-class thingProfile to super-class profile is specialized into 2 sub-classes, private and public, and has an object property, duty, that exposes a thing's profile through a set of QoT properties. These latter are represented as an object property, qualityOfThing, that has 2 data type properties, textDescription and QoTCategory.

thingGrounding. It refers to the technical specification of a thing in terms of hardware and software. Wireless sensor networks and RFID could be examples of technologies for implementing things.

Fig. 6 is the OWL-T representation of the operation dimension with focus on thingGrounding conceptual area. This representation refers to one super-class, grounding, that would encompass any "element" supporting the deployment of a thing. In this representation as well, the sub-class thingGrounding to super-class grounding has an object property, iotTechnology that captures the different deployment technologies for actuating, sensing, and communicating. These technologies are respectively with 3 sub-classes, actuator, sensor, and communicator, to iotTechnology.

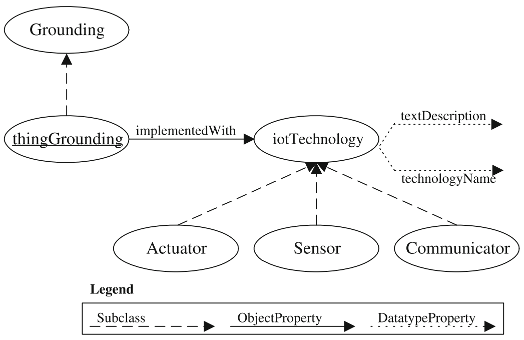


Fig. 6. OWL-T grounding representation

3.4 Consumption Dimension

The rationale of the consumption dimension is to shed light on the resources that a thing requires so that the thing functions with respect to the respective needs of the interaction and operation dimensions. To capture these resources, the consumption dimension includes one conceptual area referred to as thingResource. This latter is associated with 2 concrete areas that are *logical* and *physical* meaning that a thing could consume logical resources (r_l, e.g., solar energy) and/or physical resources (r_p, e.g., electric battery).

Building upon our previous work on resource management in business environments [7], the consumption of *logical* resources does not impact their levels of use (remain the same). The opposite happens to *physical* resources that see their levels of use decrease until they become sometimes unusable. We bind the availability of a resource to potential (sometimes concurrent) consumers to one of the below 3 cases that are *limited* (availability means that the consumption of a resource is restricted to a particular quantity and/or time period), *renewable* (availability means that the consumption of a resource continues to happen because the time period has been extended and/or an additional quantity has been provided), and *non-shareable* (availability means that the concurrent consumption of a resource must be coordinated (e.g., one at a time)).

To ensure a complete semantic description of things in the context of resource consumption, we also consider *price* of a resource whose values would fall into 3 categories referred to as *saver*, *flex*, and *flex+*. Each category promotes a different refund and change policy prior to initiating the resource consumption. It is worth noting that a price category (e.g., *saver*) for a resource might not be available or sold-out at the discretion of this resource's provider. Briefly, *saver* is the lowest price due to no refund and no change, *flex+* is the highest price due to refund and change with no fee, and *flex* is between *saver* and *flex+* prices due to refund and change with a fee.

Figure 7 is the OWL-T representation of the consumption dimension featuring one superclass, resource, that would encompass any "element" associated with consuming resources in a thing's ecosystem. In this figure, the sub-class thingResource to super-class resource captures the different types of resources that a thing could consume. These resources are represented with 2 sub-classes referred to as logical and physical. Each thingResource has 2 data type properties, textDescription and resourceName, and 3 object properties, currentState, price, and consumptionProperty. The rest of details like types of prices and types of consumption properties are all represented as sub-classes and either as data type properties or as object properties in Fig. 7.

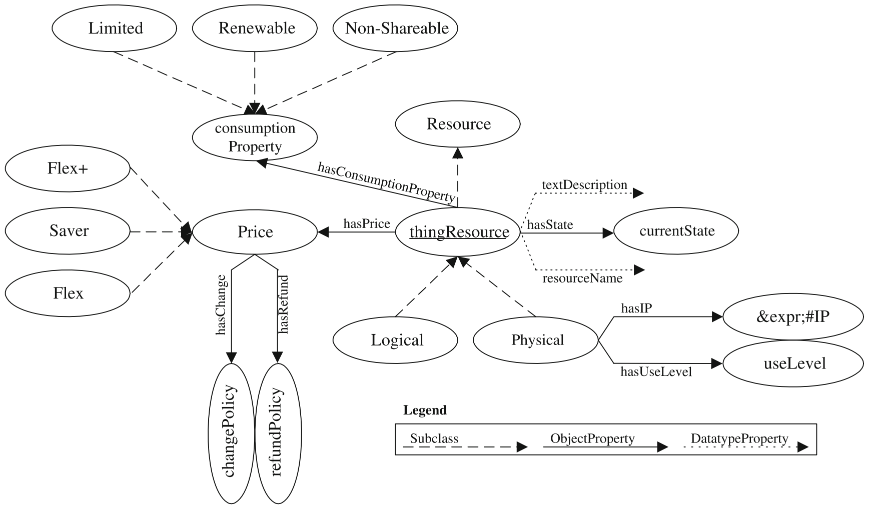


Fig. 7. OWL-T consumption representation

4 Conclusion

This paper presented a novel way for injecting semantics into things. Confined into silos, things cannot participate in complex business scenarios due to different restrictions such as lack of semantic description that would support their discovery. To address this restriction, we resorted to the well-defined standard OWL-S (Ontology Web Language for Services) to develop our OWL-T (Ontology Web Language for Things). OWL-T describes a thing from interaction, consumption, and operation dimensions. They provide a comprehensive description of things in terms of with whom they interact, what resources they consume, what they do, and how they are invoked. As future work, we would like to technically demonstrate OWL-T through a case study and examine OWL-T-based thing composition in compliance with the interaction dimension.

References

1. Agarwal, R., et al.: Unified IoT ontology to enable interoperability and federation of testbeds. In: Proceedings of WF-IoT 2016, Reston, VA, USA (2016)
2. Akshay Uttama Nambi, S.N., Sarkar, C., Venkatesha Prasad, R., Abdur Rahim Biswas, A.R.: A unified semantic knowledge base for IoT. In: Proceedings of IEEE WF-IoT 2014, Seoul, South Korea (2014)
3. Cassar, G., Barnaghi, P.M., Wang, W., Moessner, K.: A hybrid semantic matchmaker for IoT services. In: Proceedings of GreenCom 2012, Besancon, France (2012)
4. Li, K., Jiang, L.: The research of web services composition based on context in Internet of Things. In: Proceedings of CSAE 2012, Shanghai, China (2012)
5. Maamar, Z., Baker, T., Sellami, M., Asim, M., Ugljanin, E., Faci, N.: Cloud versus edge: who serves the Internet-of-Things better? Internet Technol. Lett. **1**(5) (2018)
6. Maamar, Z., Faci, N., Boukadi, K., Ugljanin, E., Sellami, M., Baker, T., Angarita, R.: How to agentify the Internet-of-Things? In: Proceedings of RCIS 2018, Rennes, France (2018)
7. Maamar, Z., Faci, N., Sakr, S., Boukhebouze, M., Barnawi, A.: Network-based social coordination of business processes. Inf. Syst. **58**, 56–74 (2016)
8. Martin, D.L., et al.: Bringing semantics to web services with OWL-S. World Wide Web **10**(3), 243–277 (2007)
9. Nicolas Seydoux, N., Khalil Drira, K., Nathalie Hernandez, N., Thierry Monteil, T.: IoT-O, a core-domain IoT ontology to represent connected devices networks. In: Proceedings of EKA 2016, Bologna, Italy (2016)
10. Qamar, A., Muhammad, A., Maamar, Z., Baker, T., Saeed, S.: A quality-of-things model for assessing the Internet-of-Thing's non-functional properties. Trans. Emerg. Telecommun. Technol. (2019, forthcoming)
11. Wang, W., De, S., Tönjes, R., Reetz, E.S., Moessner, K.: A comprehensive ontology for knowledge representation in the Internet of Things. In: Proceedings of TrustCom 2012, Liverpool, United Kingdom (2012)

OffStreamNG: Partial Stream Hybrid Graph Edge Partitioning Based on Neighborhood Expansion and Greedy Heuristic

Tewodros Ayall[1], Hancong Duan[1](✉), Changhong Liu[1], Fantahun Gereme[2], and Mesay Deleli[3]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
meettedy2123@gmail.com, duanhancong@uestc.edu.cn, 314979677@qq.com

[2] Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, Chengdu, China
fantishb@gmail.com

[3] School of Information Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
mesay_adinew@yahoo.com

Abstract. Recently, graph edge partitioning has shown better partitioning quality than the vertex graph partitioning for the skewed degree distribution of real-world graph data. Graph edge partitioning can be classified as stream and offline. The stream edge partitioning approach supports a big graph partitioning; however, it has lower partitioning quality, is affected by stream order, and it has taken much time to make partitioning compared with the offline edge partitioning. Conversely, the offline edge partitioning approach has better partitioning quality than stream edge partitioning; however, it does not support big graph partitioning. In this study, we propose partial stream hybrid graph edge partitioning OffStreamNG, which leverages the advantage of both offline and stream edge partitioning approaches by interconnecting via saved partition state layer. The OffStreamNG holds vertex and load states as partition state, while the offline component is partitioning using neighborhood expansion heuristic. And it is transferring this partition state to the online component of Greedy heuristic with minor modification of both algorithms. Experimental results show that OffStreamNG achieves attractive results in terms of replication factor, load balance, and total partitioning time.

Keywords: Edge partitioning · Stream approach · Offline approach · Distributed graph computing · Hybrid edge partitioning · Saved partition state

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 118–128, 2020.
https://doi.org/10.1007/978-3-030-54623-6_11

1 Introduction

Computing big graph data is nontrivial on a single machine, because of the memory constraint and requires much time to compute the whole input graph. Hence, the best way to process big graphs is using distributed graph processing systems such as Powerlyra [4], Powergraph [5] and Pregel [9]. In all cases, graph partitioning is one of the main component. To computing a big graph in a distributed environment, a graph should be partitioned and distributed into different clusters.

Graph partitioning is a technique to divide a big graph into smaller subgraphs based on different partitioning methods. It is a well-known NP-hard problem [2] to get an optimal solution because it is nontrivial to achieve a minimum cut ratio and maximum load balance. In general, graph partitioning is categorized into two groups, vertex and edge partitioning. Vertex partitioning is also known as edge cut. It divides a big graph into a smaller subgraph by assigning a vertex into the different partition set while considering a minimum edge cut and maximum load balance. These cut edges can act as a bridge to communicate with other partitions. Metis [6], and LDG [12] are some examples of vertex partitioners. Edge partitioning is also known as vertex cut. It divides a big graph into smaller subgraphs by assigning the edge into the different partition set while considering a minimum vertex cut and maximum load balance. These cut vertices can act as a bridge communicator between the partitions. Edge partitioners include Greedy [5], HDRF [11], DBH [13], and NE [15]. The edge partitioners have shown better partitioning quality than vertex partitioners for power-law graph [5], which very few vertices have higher degree, and many vertices have lower degree. Both partitioning methods can further be classified into two as stream and offline approaches.

Stanton and Kliot [12] proposed a stream-based approach for big graph partitioning. The stream-based partitioners ingest vertices or edges as a stream. It applies partitioning decisions on the fly based on partial knowledge of the input graph. The graph data may arrives to the partitioners in Random, Depth First Search (DFS), or Breadth First Search (BFS) order. These arrival orders affect the performance of the stream partitioners [1,3]. Offline partitioners sequentially scan the graph data and store to memory before it makes partitioning.

Stream-based edge partitioners assign a single edge at a time to the partitions based on different techniques. Hashing randomly allocates edges to the partitions based on its hash values. DBH [13] assigns the incoming edges based on the degree information of vertex. It compares the degree of the paired value of edge vertices and gives a hash value of the vertex with a smaller degree to the edge. Greedy partitioning algorithm [5] assigns the incoming edges by checking previously allocated partition state and considering a minimum load balance among each partition. Higher degree replicated first (HDRF) [11] is an edge partitioning algorithm that leverages the advantage of Greedy and adds degree information. It replicates the higher degree first and assigns the incoming edge based on a maximize HDRF computing value. Among stream edge partitioners, Hashing and DBH have a very fast running time; however, they have lower partitioning quality. On the other hand, Greedy and HDRF have a good partition quality in terms of replication factor and load balance compared with Hashing and DBH; however, they have more running time and are affected by stream order. In general, stream edge partitioners support a big graph partitioning. However, they have lower partitioning quality; require much time to make partitioning and are affected by stream orders compared with offline edge partitioning [15]. NE [15] is an offline edge partitioning and stores all input graph data to memory, then it is iteratively partitioning based on neighborhood relations. It has the best partitioning quality than the stream edge partitioners in terms of replication factor and total partitioning time; however, it does not support a big graph partitioning [15]. In this study, we propose a hybrid graph edge partitioning to improve partitioning quality and reduce the effect of stream order by taking benefits of both stream and offline partitioning approaches via stored partition state. The contributions of this work are as follows:

- We propose partial stream graph edge partitioning OffStreamNG, which uses neighborhood expansion (NE) and Greedy heuristic algorithms for the offline and stream approaches, respectively.
- We introduce the concept of holding and transferring partition state from the offline to stream partitioner with a minor modification of both algorithms.
- We experimentally check the proposed method replication factor, load balance and total partitioning time on real-world graph datasets.

This paper is organized as follows: Section 2 defines graph edge partitioning (vertex cut) problem and Sect. 3 presents the proposed method. Section 4 describes the experimental analysis and results. The conclusion is presented in Sect. 5.

2 The Graph Edge Partitioning (Vertex Cut) Problem

A given undirected graph G defined as $G = (V, E)$, where V is the set of vertices and E is the set of edges, and the size of V and E denoted as $|V| = n_v$ and $|E| = n_e$, respectively. Balanced $p-$way edge graph partitioning problem is defined as, graph G is partitioned into p partitions. Each partition has an edge set $E_k (k \in \{1, 2, ...p\})$. The edge set of each partition is not duplicated, i.e, $E_i \cap E_j = \emptyset$, where $(i, j \in \{1, 2, ...p\}, i \neq j)$.

The graph edge partitioning problem considers two factors: (i) The number of replicas (copy) vertex across partitions are minimized. (ii) The number of edges across the partitions are balanced. Let $P(v)$ be the set of partitions that each vertex $v \in V$ is replicated. Therefore, $|P(v)|$ is size of partitions that stores v. The optimization problem of $p-$way edge partition is defined by Eq. 1.

$$\min_{P} \frac{1}{n_v} \sum_{v \in V} |P(v)| . \quad \text{s.t.} \max_{k \in p} |E_k| < \epsilon \frac{n_e}{|p|} . \tag{1}$$

where $|E_k|$ and $|p|$ are the size of the edge set of the partition and the number of partitions, respectively. And $\epsilon \geq 1$ is imbalance factor. The performance of graph edge partitioning can be measured in terms of replication factor (RF), load balance and total partitioning time. Replication factor is an average of vertex replicated in each partition, as given by Eq. 2a. Load balance indicates how fairly edges are distributed in each partition and can be measured by Load relative standard deviation (LRSD), as given by Eq. 2b. Total Partitioning Time (TPT) is the summation of the ingress time (loading time of the input graph) and the running time (the time required for partitioning) of the algorithm.

$$\text{(a) } RF = \frac{1}{n_v} \sum_{i \in p} |P_i(v)|. \quad \text{(b) } LRSD = \frac{\sqrt{(\sum_{k=1}^{p} \frac{|E_k|}{\frac{n_e}{|p|}} - 1)^2 \frac{1}{|p|}}}{\frac{n_e}{|p|}}. \tag{2}$$

3 The Proposed Method

We propose partial stream graph edge partitioning based on neighborhood expansion (NE) and Greedy heuristic with minor modification of both algorithms, and it is called OffStreamNG. The OffStreamNG is the hybrid of NE and Greedy algorithms via stored partition state. Figure 1 shows the architecture of the model. The OffStreamNG model has four sub-components, Modified-NE for offline component, Modified-Greedy for online component, partition state which contains vertex and load states, and input graph splitter. Initially, the input graph is randomly split into two equal parts and is fed into the individual components. While the Modified-NE component is partitioning its input graph data, it is holding the partition state as vertex and load states. On the other hand, the Modified-Greedy component is accepting the other half of the graph data and the partition state as an input to start partitioning. The partition state is continuously accessed and updated by Modified-Greedy to allocate the incoming edges. This partition state is meant to help improve the partitioning quality of the OffStreamNG partitioner.

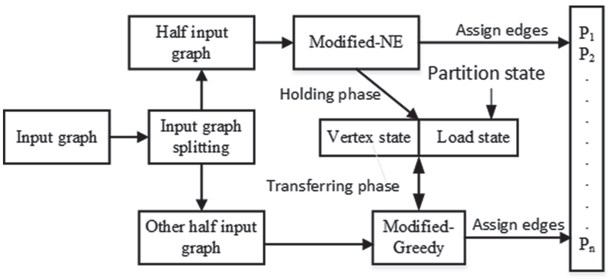
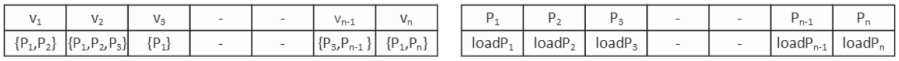


Fig. 1. Architecture of the OffStreamNG model.

3.1 Partition State

A proposed partition state is an intermediate layer of the OffStreamNG partitioner. It is recorded, while the offline component is partitioned its input graph. This partition state is stored in the main memory and accessed by the online partitioner. The partition state gives additional information to the online partitioner to identify appropriate partitions to allocate the incoming edges. The partition state has two states, vertex state, and load state, as depicted in Fig. 2. Figure 2a depicts the vertex state, which holds vertex-ids and partition set (contains all partitions in which a vertex is replicated). Figure 2b shows Load-state, which contains partition-ids and its load balance.



V_1	V_2	V_3	-	-	V_{n-1}	V_n
$\{P_1,P_2\}$	$\{P_1,P_2,P_3\}$	$\{P_1\}$	-	-	$\{P_3,P_{n-1}\}$	$\{P_1,P_n\}$

(a) Vertex state.

P_1	P_2	P_3	-	-	P_{n-1}	P_n
loadP$_1$	loadP$_2$	loadP$_3$	-	-	loadP$_{n-1}$	loadP$_n$

(b) Load state.

Fig. 2. Data structure of partition state: (a) vertex state holds vertex-ids and partition-sets. (b) load state contains partition-ids and its load balance.

3.2 The Offline Component of OffStreamNG Model

The OffStreamNG model has an offline component. This offline component uses Modified-NE and accepts half part of the input graph data. This input graph is partitioned and the partition state is saved by using Modified-NE algorithm. NE [15] is an offline edge partitioning based on neighborhood expansion heuristic. It stores all input graph data to memory and is iteratively partitioning it by growing the core set of vertices via the neighborhood relations. The NE algorithm has got two-component algorithms, edge generation, and edge allocation. In this work, we used the edge generation algorithm as described Algorithm 1 as it is while we have modified the edge allocation algorithm as described in Algorithm 2. The primary purpose of the modification is to enable the algorithm to hold vertex and load states to be used by the online component of our model.

The NE algorithm is iteratively partitioning the graph in p round. In each round k, edge set E_k is selected from the graph. Initially, it is empty edge set. Thus, E_k is expanded in steps until $|E_k| \geq \epsilon\frac{n_e}{p}$. In each round, one vertex y is randomly picked based on neighborhood expansion. The adjacent edges of y is added to E_k and y added to core set C_s. Boundary set $B_s = V(E_k)$, where $V(E_k)$ is the vertex set covered by E_k.

The main objective is to minimize the number of y added into a boundary set based on neighborhood expansion. If $B_s \backslash C_s = \emptyset$ then y is randomly selected from $V \backslash C_s$. Otherwise it is chose based on Eq. 3.

$$y = \underset{v \in B_s \backslash C_s}{\operatorname{argmin}} \mid N(v) \backslash B_s \mid . \tag{3}$$

where $|N(v) \setminus B_s|$ is the number of vertices that will be allocated to the partition k, if y is chose as C_s and its adjacent edges added to E_k.

Algorithm 1. Generate one edge partition E_k.

```
 1: Input: $E = E/2, p$
 2: Output: $E$ is allocated to $p$
 3: procedure EXPAND($E, \eta$)                                  ▷ $\eta = \epsilon \frac{n_e}{p}$
 4:     $C_s, B_s, E_k \leftarrow \emptyset$
 5:     while $|E_k| \leq \eta$ do
 6:         if $B_s \setminus C_s = \emptyset$ then
 7:             $y$ is randomly selected in $V \setminus C_s$
 8:         else
 9:             $y \leftarrow \mathrm{argmin}_{v \in B_s \setminus C_s} \mid N(v) \setminus B_s \mid$
10:         end if
11:         $ASSIGNEDGE(C_s, B_s, E_k, y)$
12:     end while
13: end procedure
```

3.3 The Online Component of OffStreamNG

The offline and online components of our model receive their corresponding graph data portions from the input graph splitter. While the offline component is partitioning its portion, it also is saving vertex and load states which is fed to the online component. We use the Greedy algorithm to build up the online component of our OffStreamNG model with minor modification on it. Greedy [5] is an online edge partitioning algorithm which improves the randomly allocated edges partition based on a heuristic. It is a Greedy sequential heuristic that places the incoming edge to the partitions based on the previously allocated partition state to minimize the expected replication factor.

Let P_{vs} and P_{ls} are vertex state and load state of the partitions, respectively. And minLoad$(P_{vs}(V))$ method returns the minimum loaded partition id from the set of $P_{vs}(V)$, where $e = (u, v)| \; u, \; v \in V$. This algorithm assigns the edge e based on the following rules:

Rule 1: If $P_{vs}(u) \cap P_{vs}(v) \neq \emptyset$, then the edge should be allocated to a partition with a minimum load in $P_{vs}(u) \cap P_{vs}(v)$.

Rule 2: If $P_{vs}(u) \cap P_{vs}(v) = \emptyset$ and $P_{vs}(u) \cup P_{vs}(v) \neq \emptyset$, then the edge should be allocated to one of the partition with a minimum load in $P_{vs}(u) \cup P_{vs}(v)$.

Algorithm 2. Modified-Edge Allocation

```
1: $P_{ls}, P_{vs} \leftarrow \emptyset$
2: procedure AssignEdge($C_s, B_s, E_k, y$)
3:     $C_s \leftarrow C_s \cup \{y\}$, $B_s \leftarrow B_s \cup \{y\}$
4:     for $a \in N(y) \setminus B_s$ do
5:         $B_s \leftarrow B_s \cup \{a\}$
6:         for $b \in N(a) \cap B_s$ do
7:             $E_k = E_k \cup \{e_{b,a}\}$
8:             $E \leftarrow E \setminus E_k$                    ▷ Holding partition state
9:             $u = e.b, v = e.a$
10:            $P_{vs}.addVertexState(u, k)$
11:            $P_{vs}.addVertexState(v, k)$
12:            if $|E_k| > \eta$ then
13:                $P_{ls}.addLoadState(|E_k|)$
14:                return
15:            end if
16:        end for
17:    end for
18: end procedure
19: procedure getVertexState()
20:    return $P_{vs}$
21: end procedure
22: procedure GetLoadState()
23:    return $P_{ls}$
24: end procedure
```

Rule 3: If only one of the two end edge vertices already has been allocated, then select a partition from the allocated vertex with minimum P_{ls}.

Rule 4: If neither u nor v have been allocated, then the edge is assigned in the partition with the least load of P_{ls}.

However, while the partition state information is very important for the Greedy algorithm to make a decision, it has minimal information at the beginning, which makes the partition quality relatively weak. In this work, we have made a minor modification on the Greedy algorithm. The Modified-Greedy describes in Algorithm refalgo:phasetwo, which takes rich partition state information from Modified-NE algorithm. By getting more partition state information from the offline component, the online component gets sharpened in decision making.

Algorithm 3. Modified-Greedy

```
 1: Input:$E = E/2$, $p$
 2: Output: Generate a $partition\_Id$ where $E$ to be allocated.
 3: $P_{vs} \leftarrow GETVERTEXSTATE()$                    ▷ Accessing the partition state
 4: $P_{ls} \leftarrow GETLOADSTATE()$
 5: procedure GETPARTITIONID($e$, $p$, $P_{vs}$, $P_{ls}$)
 6:     $u = e.u, v = e.v$
 7:     if $P_{vs}(u) \cap P_{vs}(v) \neq \emptyset$ then
 8:         $partition\_Id = minLoad(P_{vs}(u) \cap P_{vs}(v))$
 9:     else if $P_{vs}(u) \cap P_{vs}(v) = \emptyset$ && $P_{vs}(u) \cup P_{vs}(v) \neq \emptyset$ then
10:         $partition\_Id = minLoad(P_{vs}(u) \cup P_{vs}(v))$
11:     else if $P_{vs}(u) = \emptyset$ && $P_{vs}(v) \neq \emptyset$ then
12:         $partition\_Id = minLoad(P_{vs}(v))$
13:     else if $P_{vs}(u) \neq \emptyset$ && $P_{vs}(v) = \emptyset$ then
14:         $partition\_Id = minLoad(P_{vs}(u))$
15:     else if $P_{vs}(u) = \emptyset$ && $P_{vs}(v) = \emptyset$ then
16:         $partition\_Id = minLoad(P_{ls})$
17:     end if
18:     return $partition\_Id$
19: end procedure
```

4 Experimental Analysis and Results

We implemented OffStreamNG partitioner in an 8 core CPU Ubuntu machine with 64 GB memory. For comparison purpose, we used open-source implementation of edgepart[1] and VGP[2] for NE and stream(Hashing, DBH, Greedy, and HDRF), respectively. We used imbalance factor $\epsilon = 1.1$ and for HDRF $\lambda = 1.1$. We used real-world edge list graph datasets, com-Livejournal from SNAP [8] and Orkut from KONECT [7]. These datasets are randomly ordered. Table 1 shows the characteristics of datasets.

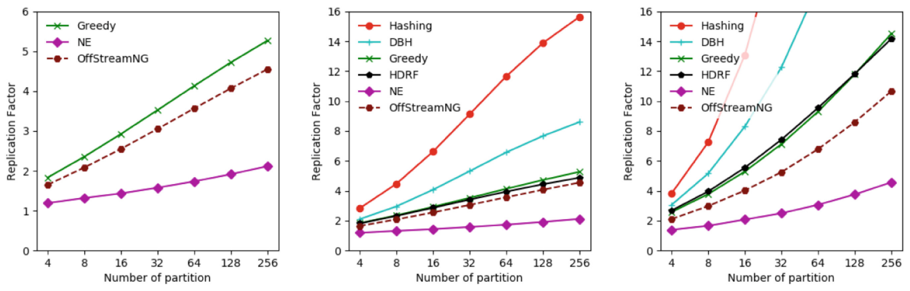
Table 1. Real world graph datasets.

Dataset	n_v	n_e
Com-Livejournal [14]	5,203,764	48,708,948
Orkut [10]	3,072,441	117,184,899

4.1 Experimental Results

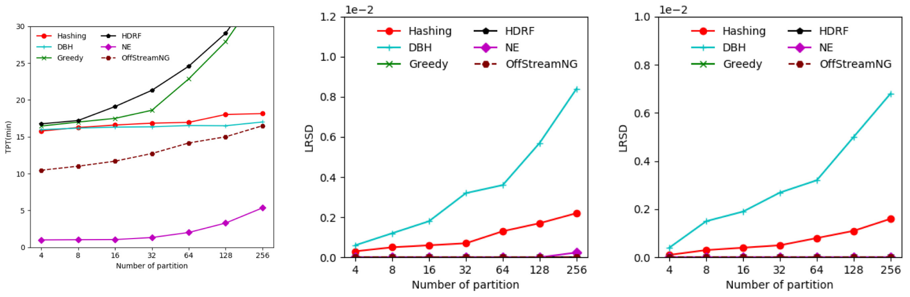
Series of experiments were conducted, and results were carefully recorded. Comparative result analysis is made using the evaluation metrics. Figure 3 shows the replication factor and Fig. 4 shows load balance of com-Livejournal and Orkut datasets.

[1] https://github.com/ansrlab/edgepart.
[2] https://github.com/fabiopetroni/VGP.



(a) Boundedness of RF value. (b) Com-Livejournal. (c) Orkut.

Fig. 3. Replication factor against the number of target partitions (log-log scale) on real-world graph datasets. (a) it shows boundedeness of RF value in Com-Livejournal dataset.



(a) TPT. (b) Com-Livejournal. (c) Orkut.

Fig. 4. Total partitioning time (TPT) and load balance against the number of target partitions (log-log scale). (a) it shows TPT of com-Livejournal. (b) and (c) show load balance.

4.2 Discussion

We evaluate the performance of OffStreamNG by measuring the following metrics:

Replication Factor (RF): RF value is calculated using Eq. 2a and is depicted in Fig. 3. We compared the performance of our OffStreamNG in terms of RF among online edge partitioners such as DBH, Greedy, and HDRF. And also with offline edge partitioner, NE, on real-world graph datasets with a set of target partitions [4, 8, 16, 32, 64, 128, 256]. We calculated average RF from individual RF values on each partition and further averaged these values for all datasets considered. Comparing the calculated average RF values, OffStreamNG performed 62% lower than Hashing, 46% lower than DBH, 20% smaller than Greedy, 18% smaller than HDRF. The RF value showed that OffStreamNG performed far better than the stream edge partitioner. The RF value of OffStreamNG is smaller than other algorithms because it gets more partition state from the offline component to make a better decision. Generally, Fig. 3a shows that the replication factor (RF) of OffStreamNG bounds between the pure offline and online partitioners.

Load balance: We measured the load balance by LRSD as given by Eq. 2b. The load balance is illustrated in Fig. 4b and Fig. 4c for Com-Livejournal and Orkut, respectively. The curves show that HDRF, Greedy, NE and OffStreamNG performed best as the number of partition grows. Hashing and DBH are the worst performers as load skew grows as the number of target partitions grows.

Total Partitioning Time: We compared OffStreamNG among the stream edge partitioners and the offline NE as shown Fig. 4a on Com-LiveJournal dataset with the number of partitions ranging 4 to 256. The result shows that our OffStreamNG partitioner scored an average TPT improvement of 20% smaller than DBH, 23% smaller than Hashing, 38% smaller than Greedy and 43% smaller than HDRF. Expectedly, NE has smaller TPT than our hybrid partitioners because OffStreamNG is partial streaming. The overall results showed that OffStreamNG scored lower TPT compared with the state of the art stream based partitioners.

5 Conclusion

Graph edge partitioning has dramatically determined the performance of distributed graph processing systems in terms of communication and workload costs. In this study, we proposed partial stream graph edge partitioning OffStreamNG by leveraging both the offline and stream edge partitioning approaches by introducing the concept of holding partition state from the offline and transferring this state to the online partitioner. The OffStreamNG uses neighborhood expansion (NE) and Greedy heuristic for the offline and online components with minor modification of both algorithms, respectively. We compared OffStreamNG with edge partitioners, which OffStreamNG scores diminished value of the replication factor, the optimum load balance, and good total partitioning time.

References

1. Abbas, Z., Kalavri, V., Carbone, P., Vlassov, V.: Streaming graph partitioning: an experimental study. Proc. VLDB Endow. **11**(11), 1590–1603 (2018)
2. Andreev, K., Racke, H.: Balanced graph partitioning. Theory Comput. Syst. **39**(6), 929–939 (2006). https://doi.org/10.1007/s00224-006-1350-7
3. Ayall, T., Duan, H., Liu, C.: Edge property based stream order reduce the performance of stream edge graph partition. J. Phys. Conf. Ser. **1395**, 012010 (2019). IOP Publishing
4. Chen, R., Shi, J., Chen, Y., Zang, B., Guan, H., Chen, H.: PowerLyra: differentiated graph computation and partitioning on skewed graphs. ACM Trans. Parallel Comput. (TOPC) **5**(3), 13 (2019)

5. Gonzalez, J.E., Low, Y., Gu, H., Bickson, D., Guestrin, C.: PowerGraph: distributed graph-parallel computation on natural graphs. In: Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12), pp. 17–30 (2012)
6. Karypis, G.: METIS: unstructured graph partitioning and sparse matrix ordering system. Technical report (1997)
7. Kunegis, J.: Konect: the koblenz network collection. In: Proceedings of the 22nd International Conference on World Wide Web. pp. 1343–1350. ACM (2013)
8. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection, June 2014. http://snap.stanford.edu/data
9. Malewicz, G., et al.: Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 135–146. ACM (2010)
10. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proceedings of the 7th ACM SIGCOMM conference on Internet Measurement, pp. 29–42. ACM (2007)
11. Petroni, F., Querzoni, L., Daudjee, K., Kamali, S., Iacoboni, G.: HDRF: stream-based partitioning for power-law graphs. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 243–252. ACM (2015)
12. Stanton, I., Kliot, G.: Streaming graph partitioning for large distributed graphs. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1222–1230. ACM (2012)
13. Xie, C., Yan, L., Li, W.J., Zhang, Z.: Distributed power-law graph computing: theoretical and empirical analysis. In: Advances in Neural Information Processing Systems, pp. 1673–1681 (2014)
14. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. Knowl. Inf. Syst. **42**(1), 181–213 (2013). https://doi.org/10.1007/s10115-013-0693-z
15. Zhang, C., Wei, F., Liu, Q., Tang, Z.G., Li, Z.: Graph edge partitioning via neighborhood heuristic. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 605–614. ACM (2017)

Temporal Enrichment and Querying of Ontology-Compliant Data

Jing Ao[1(✉)], Zehui Cheng[2], Rada Chirkova[1], and Phokion G. Kolaitis[2]

[1] NC State University, Raleigh, NC 27695, USA
{jao,rychirko}@ncsu.edu
[2] UC Santa Cruz, Santa Cruz, CA 95064, USA
{zecheng,kolaitis}@ucsc.edu

Abstract. We consider the problem of answering temporal queries on RDF stores, in the presence of time-agnostic RDFS domain ontologies, of relational data sources that include temporal information, and of rules that map the domain information in the source into the target ontology. Our proposed solution consists of two rule-based domain-independent algorithms. The first algorithm materializes target RDF data via a version of data exchange that enriches the data and the ontology with temporal information from the sources. The second algorithm accepts as inputs temporal queries expressed in terms of the domain ontology, using SPARQL supplemented with time annotations. The algorithm translates the queries into the standard SPARQL form that respects the structure of the temporal RDF information while preserving the question semantics. We present the algorithms, report on their implementation and experimental results for two application domains, and discuss future.

Keywords: Data-intensive sciences and databases · Temporal databases · Data exchange · RDF/RDFS/SPARQL

1 Introduction

In application domains that span industry, government, science, and global health, data are often collected independently by different teams over time. As the needs of the various data-collecting entities evolve, it is often the case that data from multiple *sources* must be put together under a unified *target* format *(exchanged* [1]), using expert-developed *source-to-target (s-t) rules.* In many applications, the target data formats also have to be aligned with the standard domain vocabularies called *ontologies.* Our exposition will focus on a common real-life scenario, in which ontologies and ontology-compliant data are expressed using the *RDF/S* capabilities – those of the Resource Description Framework *(RDF)* data model [2] enriched with additional *RDFS* specifications [3], – and are queried using SPARQL [4], while the source data are relational.

In applications conforming to this relational-to-RDF/S data-exchange scenario, e.g., in studies of antimicrobial resistance *(AMR)*, the source data may contain important temporal information, while the applicable target domain ontologies lack temporal components. (In AMR this is the case with the Antibiotic Resistance Ontology *ARO*). Existing relational-to-RDF/S data-exchange solutions do not directly apply here, as they do not incorporate temporal semantics of the data in easy-to-use ways. As a result, temporal information from the sources can be lost in the exchange process, making it hard or even impossible for domain scientists to efficiently obtain correct answers to temporal queries posed on the contents of the source data in terms of the target ontologies. Custom solutions developed on a case-by-case basis [5] would delegate to data analysts or domain scientists the nontrivial task of temporally enhancing the originally time-agnostic domain ontologies, such as ARO. In addition, to correctly formulate temporal queries, domain analysts would need to be aware of how the temporal information is modeled and represented in the resulting systems.

Contributions. In this paper, in the context of relational-to-RDF/S data exchange, we consider the scenario in which domain analysts are interested in obtaining answers to temporal queries formulated in terms of the given time-agnostic target domain ontology, with the expectation that the temporal information in the query answers would come from the data sources. We assume that the analysts (users) are familiar with formulating SPARQL queries using the given RDFS ontology, and that they provide the s-t rules that map the domain information in the source schemas into the time-agnostic target ontology, using tools such as that of [11]. In this scenario, we propose a declarative domain-independent approach that enables users to formulate SPARQL-based temporal queries and returns to them answers to the queries, using the domain information enabled in the target by the s-t rules, with the temporal dimension of that information coming from the sources via temporal enrichment.

Our approach focuses on separating temporal semantics from the domain semantics, and comprises two algorithms. The first algorithm materializes target RDF data via a version of data exchange that builds on the given s-t rules to enrich the target data and ontology with temporal information from the sources. The second algorithm accepts as inputs temporal queries expressed in terms of the ontology, using SPARQL supplemented with a lightweight formalism for time annotations and comparisons. The algorithm translates queries into the standard SPARQL form that respects the structure of the temporal RDF information while preserving the question semantics, thus ensuring successful evaluation of the queries on the materialized temporally-enriched RDF data. In this paper we present the algorithms (Sect. 2–3), report on their implementation and experimental results for two application domains (Sect. 4), and discuss future work (Sect. 5). Please see the full version of the paper [18] for the details.

Related Work. RDFS [3] is a language used in practice for describing ontologies. Existing works have focused on representing and reasoning with temporal RDF data [6], querying such data [7], and inferring temporal properties in temporal RDFS ontologies [8]. At the same time, the temporal aspect is usually not included in the practical development of domain ontologies; our proposed approach in this paper is designed to bridge this gap.

Relational data exchange has been studied extensively [1]. For relational-to-RDF data exchange, see [10,11]. To the best of our knowledge, temporal data exchange between relational schemas and ontologies has not been studied formally. The only formal work on temporal relational data exchange is in [12]. We use the results of [12] in the experimental validation of our approach.

2 Temporal Enrichment of Ontologies and Data

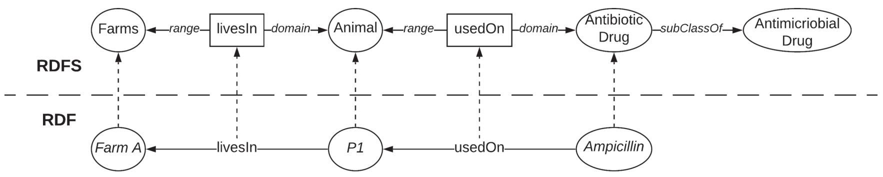


Fig. 1. The RDF (lower) level of this Figure shows two "subject-predicate-object" `(s,p,o)` triples, with names (URIs) of resources (e.g., *Farm A*), and predicate names (e.g., *livesIn*). At the RDFS level, the classes of the entities are related to each other through the domains and ranges of the predicates. Class *Antibiotic Drug* is shown to be a `subclassOf` *Antimicrobial Drug.* The two layers are connected via `type` statements.

The first problem that we consider is enrichment of time-agnostic RDFS ontologies and of the resulting materialized RDF data with temporal information from the relational sources. Our domain-independent rule-based Algorithm 1, which addresses the problem, accepts three inputs. The first input comprises *relational data sources* with temporal information. We assume that temporal information in a relation, if present, is expressed via a single *marked* column whose values are time intervals. (Specifically, we assume *concrete* representation of *valid time* [13].) The second input is the target time-agnostic *RDFS domain ontology.* The final input is a set of *source-to-target tuple-generating dependencies (s-t tgds)* expressing the rules by which the source *domain* data can be materialized *(exchanged)* in the format conforming to the target ontology. We assume that each rule is a *GLAV s-t tgd* [1] with up to one temporal variable, which (if present) occurs once on the left-hand side *(LHS)* [12]. For s-t tgds to make sense in the relational-to-RDF/S scenario, we represent each RDF/S triple on the right-hand side *(RHS)* of the tgds, of the form "subject-predicate-object," or (s, p, o), as a relational atom of the form $p(s, o)$.

Algorithm 1 is based on straightforward domain-independent pattern-based rules, and can be viewed as consisting of three conceptually distinct stages. In the first stage, the algorithm adds "temporal-enrichment atom patterns" to the RHS of the input s-t tgds. For the patterns, we use the temporal structures of [6], which, essentially, reify [14] RDF triples with their relevant temporal adornments, see the RDF level of Fig. 2 for an illustration. (We use the structural patterns of [6] to allow use of graph DBMSs without any special features for storing the RDF results of materializing temporal data from the sources.) In

Algorithm 1: Temporally enriching ontologies, s-t tgds, and RDF data

```
Data: Relational data sources $\mathcal{D}$, RDFS ontology $\mathcal{O}$, and set $\mathcal{M}$ of s-t tgds.
Result: Temporally enriched $\mathcal{O}^T$, $\mathcal{M}^T$, and target RDF data set $\mathcal{F}^T$.
begin
    $\mathcal{M}^T \leftarrow \mathcal{M}$; $\mathcal{O}^T \leftarrow \mathcal{O}$; // initialization
    for each atom $p(s, o)$ on the right-hand side of each $M \in \mathcal{M}$ do
        if $p(s, o)$ is in the temporal-enrichment scope of $M$ then
            $\mathcal{M}^T \leftarrow$ temporally enrich $p(s, o)$ in $M$; // first stage
            $\mathcal{O}^T \leftarrow$ temporally enrich the $p$-related part of $\mathcal{O}^T$; // second stage
    $\mathcal{F}^T \leftarrow$ materialize $\mathcal{D}$ into RDF via data exchange using $\mathcal{M}^T$; // third
        stage
    return $\mathcal{O}^T$, $\mathcal{M}^T$, and $\mathcal{F}^T$;
```

the second stage, the input time-agnostic ontology is augmented with RDFS-level specifications of the temporal-enrichment structures that enriched the s-t tgds. In the third stage, the resulting s-t tgds can be used to exchange the input (temporally aware) data sources into the temporally aware RDF format consistent with the (now) temporally aware output ontology. (We assume that all of the materialized RDF data conform to the enriched ontology).

Consider an example in the AMR domain. Suppose a data source has a relation *DrugUsage (Farm, Animal, AMR-Drug, Drug-Administration-Time*) for recording the temporal history of AMR drug usage for animals in farms. Let the relation have a single tuple *('Farm A','P1','Ampicillin', [1/1/2019,1/5/2019]).* Suppose that analysts would like to obtain answers to temporal queries posed using the ontology terminology shown at the RDFS (top) level of Fig. 1. As the ontology is time agnostic, the best way to exchange data from the *DrugUsage* source to a target consistent with the ontology would be to use the s-t tgd.

$$DrugUsage(f, a, d, \underline{\mathbf{t}}) \rightarrow livesIn(a, f) \wedge usedOn(d, a). \tag{1}$$

Here, $\underline{\mathbf{t}}$ is a temporal variable for the temporal attribute. Using this s-t tgd on the *DrugUsage* relation would result in the data shown at the RDF level of Fig. 1. Clearly, AMR scientists cannot get from these data a correct (nonempty) answer to the query "return the farms that used antibiotic drugs on their animals in the year 2019," as there is no temporal information in the stored data of Fig. 1.

This problem can be solved by applying Algorithm 1 to the above ontology, data source, and s-t tgd inputs. The algorithm will yield the enriched s-t tgd.

$$\begin{aligned} DrugUsage(f, a, d, \underline{\mathbf{t}}) \rightarrow\ & livesIn(a, f) \wedge usedOn(d, a) \wedge tsubj(c_1, d) \\ & \wedge\ tpred(c_1, \texttt{usedOn}) \wedge tobj(c_1, a) \wedge temporal(c_1, c_2) \\ & \wedge\ interval(c_2, c_3) \wedge validFor(c_3, \underline{\mathbf{t}}). \end{aligned} \tag{2}$$

The RHS of Eq. (2) exhibits the temporal structure of [6] applied to the RDF triple represented by the atom $usedOn(d, a)$. c_1 (c_2, c_3, resp.) stands for unique

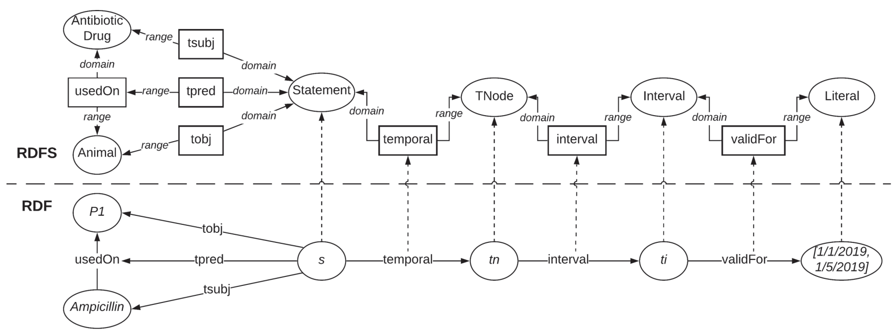


Fig. 2. An adornment of the *Ampicillin-[is]-usedOn-P1* RDF triple of Fig. 1 with a temporal structure of [6]. The RDFS layer shows the metadata of [6], including a *Statement* class and a *TNode* (temporal-node) class. The *TNode* is characterized by an *interval*-value class. The RDF level shows instantiations of these RDFS metadata.

new URIs generated for the temporal structure of [6] with the RDF triples being materialized; e.g., s, tn, and ti are generated for the triple *Ampicillin-[is]-usedOn-P1* in Fig. 2. The top half of Fig. 2 shows the time-enriched ontology information that results from applying Algorithm 1 to the inputs of the example.

3 Querying the Materialized Temporally Enriched Data

```
SELECT ?d ?f ?t
WHERE
{
    ?d    amr:usedOn   ?a      [?t].
    ?d    rdf:type          amr:AntimicrobialDrug.
    ?t    during  "[2019-01-01,2019-12-31]".
    ?a    amr:livesIn     ?f.
}
```
(a)

```
SELECT ?d ?f ?t
WHERE
{
    ?d    amr:usedOn        ?a      [?t].
    ?d    rdf:type          amr:AntibioticDrug.
    ?t    during  "[2019-01-01,2019-12-31]".
    ?a    amr:livesIn     ?f.
}
```
(b)

```
SELECT ?d ?f ?t
WHERE
{
    ?d    amr:usedOn            ?a.
    ?s    temporal:tsubj        ?d.
    ?s    temporal:tpred        amr:usedOn.
    ?s    temporal:tobj         ?a.
    ?s    temporal:temporal     ?tn.
    ?tn   temporal:interval     ?i.
    ?i    temporal:validFor     ?t.
    ?d    rdf:type          amr:AntibioticDrug.
    ?a    amr:livesIn       ?f.
    FILTER(
    intialDate(?t)>"2019-01-01"^^xsd:dateTime
    AND
    finalDate(?t)<"2019-12-31"^^xsd:dateTime).
}
```
(c)

Fig. 3. Query Q^T_{am} asking for the farms that used antimicrobial drugs in 2019, as *(a)* the original *temporally-annotated* SPARQL version, *(b)* the result of its rewriting by the 1st stage of Algorithm 2, and *(c)* the result of the expansion of version *(b)* by the 2nd stage of Algorithm 2. (In *(c)*, *initialDate* and *finalDate* are shorthand for SPARQL functions for extracting the start/end points from the time-interval values bound to *?t*.) Unlike *(a)*–*(b)*, version *(c)* is directly executable by standard SPARQL processors.

Suppose that Algorithm 1 has been applied to the given relational data sources $\mathcal{D}$, time-agnostic target ontology $\mathcal{O}$, and s-t tgds $\mathcal{M}$. As a result, we obtain an RDF/S data set $(\mathcal{F}^T, \mathcal{O}^T)$ that materializes source information, including temporal characterizations of the source data. Now the RDF query language

SPARQL [4] can be used to formulate, with respect to (w.r.t.) $(\mathcal{F}^T, \mathcal{O}^T)$, temporal queries such as Q^T_{am}: "Return farms that used antimicrobial drugs in the year 2019," see Fig. 3(c). This temporal query can be processed directly on the data set $(\mathcal{F}^T, \mathcal{O}^T)$ by a standard SPARQL processor, with a nonempty answer successfully returned on the data coming from the *DrugUsage* relation.

As illustrated in Fig. 3(c), direct temporal querying of temporal RDF/S data sets is already enabled by our approach of Sect. 2. At the same time, our additional objective is to allow domain analysts to concentrate on the domain-ontology part of formulating such temporal queries, while keeping the temporal part of the queries as easy to write as possible. For this purpose, we offer domain experts an opportunity to formulate their temporal queries via a *temporal user interface (temporal UI)* that we provide for SPARQL. In the UI, standard SPARQL constructs are supplemented with *temporal annotations* on RDF/S triple patterns in the queries, using the notation that we borrow from the query format of [8], as well as with constructs for temporal comparisons, such as `during`, which are known as *Allen's interval relations* [9]. See Fig. 3(a) for an illustration, with temporal annotation $?t$. We will be referring to temporal-UI versions of SPARQL queries as *temporally annotated SPARQL queries.*

Algorithm 2: Temporal querying of temporally enriched RDF/S data

```
Data: RDFS ontology $\mathcal{O}^T$, RDF data set $\mathcal{F}^T$, temporally annotated SPARQL
      query $Q$.
Result: Answer set $\mathcal{A}$ to a SPARQL reformulation of $Q$ on $\mathcal{F}^T$.
begin
    $\mathcal{R} \leftarrow \{Q\}$; // will reformulate $Q$ into $\mathcal{R}$ that is executable on $\mathcal{F}^T$
    for each triple pattern $P$ in $\mathcal{R}$ do
        if there is a hierarchy $H$ in $\mathcal{O}^T$ that applies to $P$ then
            $\mathcal{R} \leftarrow$ rewrite $P$ in $\mathcal{R}$ in all ways using $H$; // 1st stage: rewriting
    for each temporal annotation $T$ in $\mathcal{R}$ do
        $\mathcal{R} \leftarrow$ expand $T$ in $\mathcal{R}$ into triple patterns; // 2nd stage: expansion
    $\mathcal{A} \leftarrow \emptyset$; // initializing set of answers to $\mathcal{R}$ on RDF data set $\mathcal{F}^T$
    for each SPARQL query $R$ in $\mathcal{R}$ do
        $\mathcal{A} \leftarrow$ use SPARQL processor to add to $\mathcal{A}$ the result of processing $R$ on
        $\mathcal{F}^T$;
    return $\mathcal{A}$;
```

We now present a domain-independent approach for reformulating temporally annotated SPARQL queries into (standard) SPARQL queries that respect the structure of the temporal RDF information while preserving the semantics of the questions. Acting on top of a SPARQL processor, our Algorithm 2 ensures successful evaluation of temporally annotated SPARQL queries on the materialized temporally-enriched RDF/S data generated by Algorithm 1 (Sect. 2).

Algorithm 2 accepts as inputs RDF/S data sets $(\mathcal{F}^T, \mathcal{O}^T)$ and temporally annotated SPARQL queries Q expressed in terms of the domain-ontology part of $\mathcal{O}^T$. The algorithm reformulates each given Q into a set $\mathcal{R}$ of SPARQL queries conforming to the ontology $\mathcal{O}^T$, and then uses the SPARQL processor to obtain the answer to Q, by processing all the queries in $\mathcal{R}$ on the data set $(\mathcal{F}^T, \mathcal{O}^T)$.

The reformulation part of Algorithm 2 works in two stages, rewriting (1st stage) and expansion (2nd stage). In the 1st stage, the algorithm uses domain-independent pattern-based rules to repeatedly "unfold," in the queries being rewritten, `:subClassOf` and `:subPropertyOf` hierarchies w.r.t. the RDFS ontology $\mathcal{O}^T$ using entailment rules, see, e.g., [14]. As a result, the input query Q is turned into a set $\mathcal{R}$ of temporally annotated SPARQL queries that would be directly executable on the data set $\mathcal{F}^T$ *but for* their temporal annotations. This process would transform the query of Fig. 3(a) into the query of Fig. 3(b). The 2nd, expansion, stage of the query-reformulation process in Algorithm 2 uses domain-independent pattern-based rules to replace the temporal annotations in the queries $\mathcal{R}$ with standard RDF/S constructs. Specifically, all the temporal annotations of individual triple patterns in $\mathcal{R}$ are replaced with their structural counterparts of [6] (as in, e.g., Fig. 2), and all the Allen's interval relations (e.g., `during`) are replaced with built-in comparisons on the endpoints of the time intervals involved. (This process would transform the query of Fig. 3(b) into the query of Fig. 3(c).) The resulting SPARQL queries are submitted by the algorithm to the SPARQL processor to obtain the answers to the input query.

4 Implementation and Experimental Results

We have implemented Algorithms 1–2 on top of Java 1.8, PostgreSQL 11, and RDF4J 3.0.1, using the Llunatic [15] rule interpreter for reformulating temporally annotated queries into standard executable SPARQL queries. For the experiments, we used data environments in two application domains, AMR and TPC-BiH [17]. Each environment included a relational source schema, a time-agnostic target RDFS domain ontology and, for translating the schema into the ontology, a set of GLAV s-t tgds each with at most one temporal variable, which, if present, would occur exactly once on the LHS. Each data environment also included relational source data generated with DataFiller [16] at multiple scale factors, as well as temporal queries defined in terms of the domain ontologies.

The experiments were designed around two properties of the outcomes of applying to the AMR and TPC-BiH environments the approach of Algorithms 1–2 for temporal RDF/S enrichment and querying: (1) degree of preservation in the target of the temporal information from the sources, see Fig. 4; and (2) degree of correctness of the answers to temporal queries on the target, w.r.t. the answers obtained in the baseline relational-to-relational approach supported by the formal results of [12], see Fig. 5. We evaluated the latter property both for queries that required rewriting w.r.t. `:subClassOf` and `:subPropertyOf` hierarchies in the given ontologies (1st stage of Algorithm 2), and for queries that did not require such rewriting. (See [18] for the details of our methodology.) We also evaluated the efficiency of our implementation, see Fig. 6.

As a high-level summary of our experimental results, for each data environment used in the experiments, with each selected scale factor, and for each temporal query that was considered, the experimental results were identical between our relational-to-RDFS setting and the baseline relational-to-relational setting. (The formal correctness of the outcomes in the latter setting is supported by the results of [12].) We conclude that all these results experimentally validate the correctness of the proposed approach.

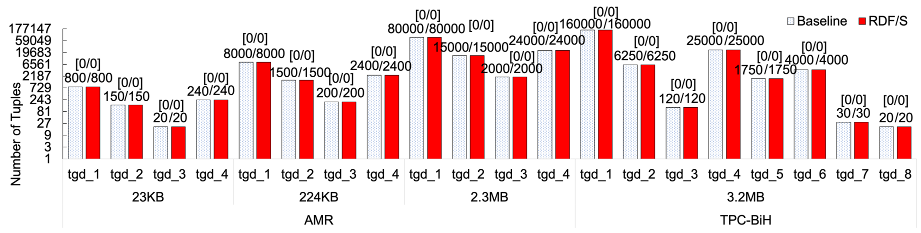


Fig. 4. Evaluating information loss in data exchange with temporal RDF/S enrichment vs. baseline outcomes. The X-axis shows the names of the s-t tgds and the source-data sizes for the environments tested; the (logarithmic) Y-axis shows the number of resulting data tuples. The $[A/B]$ notation on top of the target data-size bars shows the relative number of unmatched tuples between the two sets.

Figure 4 reports our results, in the AMR and TPC-BiH data environments, for the degree of preservation in the target of the temporal information from the sources, as enabled by Algorithm 1. For all the results, we got $A = B = 0$; that is, in each experiment we obtained the same sets of tuples in the target temporal data as in the baseline case. We conclude that the results experimentally validate the correctness of our temporal-enrichment Algorithm 1.

Figure 5 reports our results, in the AMR and TPC-BiH data environments, for the degree of correctness of the answers to temporal queries on the RDF/S target (Algorithm 2) w.r.t. the relational answers that would be obtained in the baseline approach. All the input queries were temporally annotated SPARQL queries of the form illustrated in Fig. 3(a), which were then reformulated into standard SPARQL queries via Algorithm 2, as illustrated in Fig. 3(c). We used the certain-answer semantics [1] in processing all the queries. Given that $A = B = 0$ in all cases, we conclude that our results for the degree of correctness of the answers to temporal queries on the RDF/S target experimentally validate the correctness of the proposed query-reformulation Algorithm 2.

Figure 6 reports the results for the runtime overhead of our implementation of the query-reformulation part of Algorithm 2, as part of the overall response times for the queries tested. The response times were measured both for queries that did not require rewriting w.r.t. RDFS hierarchies (1st stage of Algorithm 2), see Fig. 6(a), and for queries requiring such rewriting, see Fig. 6(b). Not surprisingly, in all the cases tested, the overhead of Algorithm 2 depended only on the size of the input query, rather than on the size of the stored data processed by

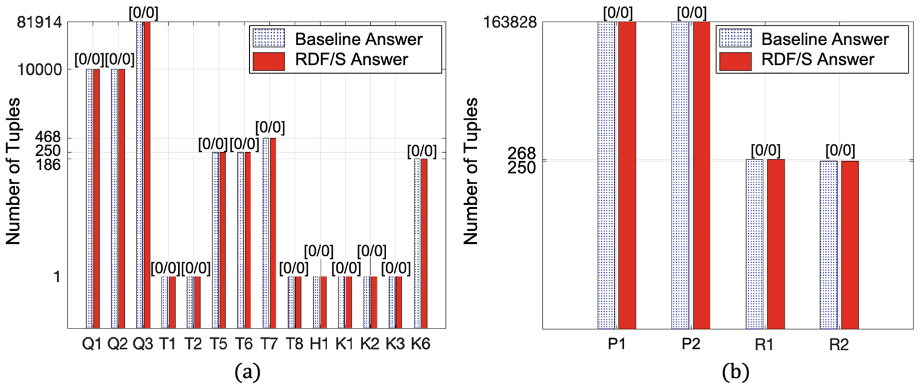


Fig. 5. Evaluating information loss in answers to temporal queries vs expected baseline outcomes. The X-axes show the names of the AMR and TPC-BiH queries tested. The (logarithmic) Y-axes show query-answer sizes in tuples. The $[A/B]$ notation on top of the bars shows the relative number of unmatched tuples between the two sets.

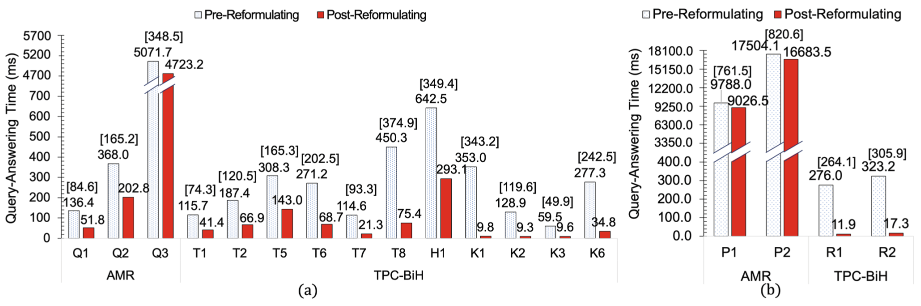


Fig. 6. Measuring the time overhead of reformulating temporally annotated queries into executable SPARQL. The X-axes show the names of the queries tested. The (logarithmic) Y-axes show the 10-runtime averaged overall response times in *ms*. The values in square brackets show the difference, for each query, between the processing time with the reformulation overhead included (left bar) and excluded (right bar).

the query, or on the size of the query answer. As a result, even for queries whose runtimes were over 16 *sec after* the reformulation part of Algorithm 2, the overhead of applying Algorithm 2 was under 821 *ms*; this value is below the user-tolerance time threshold for interactive systems [19]. We conclude that the runtime overhead of Algorithm 2 in the reformulation of temporally annotated SPARQL queries is sufficiently small to be tolerated by users.

5 Conclusions and Future Work

In this paper we considered the scenario in which domain analysts and scientists are interested in obtaining answers to *temporal* queries formulated in terms of the given *time-agnostic* RDFS domain ontology, in the presence of temporal information in relational data sources and of source-to-target (s-t) rules for mapping domain information between the sources and the target ontology. We presented our declarative domain-independent algorithmic approach to addressing the temporal-enrichment and query-answering problems in this scenario. In our report on the approach, we described the algorithms and their implementation, and presented our experimental results for two application domains.

Providing formal proofs of correctness of our proposed approach is an immediate direction of future work. Other directions of future formal and practical work on the topics discussed in this paper include incorporation into the framework of richer ontology formalisms such as OWL, as well as of data-exchange dependencies that are more expressive in their temporal aspect than those of [12]. Another promising direction of research lies in designing and developing user interfaces that would make it easier for domain scientists that are not computer experts to query their temporal data in terms of domain ontologies.

References

1. Arenas, M., Barceló, P., Libkin, L., Murlak, F.: Foundations of Data Exchange. Cambridge University Press, Cambridge (2014)
2. Hayes, P. (ed.) RDF Semantics: W3C Recommendation (2004). https://www.w3.org/TR/2004/REC-rdf-mt-20040210/
3. Brickley, D., Guha, R.V. (eds.) RDF Vocabulary Description Language: RDF Schema (2014). https://www.w3.org/TR/rdf-schema/
4. SPARQL query language for RDF. https://www.w3.org/TR/rdf-sparql-query/
5. Michel, F., Montagnat, J., Zucker, C.F.: A survey of RDB to RDF translation approaches and tools, Rapport de Recherche ISRN I3S/RR 2013–04-FR (2014)
6. Gutiérrez, C., Hurtado, C.A., Vaisman, A.A.: Introducing time into RDF. IEEE Trans. Knowl. Data Eng. **19**(2), 207–218 (2007)
7. Tappolet, J., Bernstein, A.: Applied temporal RDF: efficient temporal querying of RDF data with SPARQL. In: Proceedings of the ESWC, pp. 308–322 (2009)
8. Zimmermann, A., Lopes, N., Polleres, A., Straccia, U.: A general framework for representing, reasoning and querying with annotated Semantic Web data. J. Web Semant. **11**, 72–95 (2012)
9. Allen, J.F.: Maintaining knowledge about temporal intervals. CACM **26**, 832–843 (1983)
10. Boneva, I., Lozano, J., Staworko, S.: Relational to RDF data exchange in presence of a Shape Expression Schema. arXiv preprint arXiv:1804.11052 (2018)
11. Boneva, I., Dusart, J., Fernández-Álvarez, D., Gayo, J.E.L.: Shape designer for ShEx and SHACL constraints. In: Proceedings ISWC Satellite Tracks, pp. 269–272 (2019)
12. Golshanara, L., Chomicki, J.: Temporal data exchange. Inf. Syst. **87**, 101414 (2020)
13. Snodgrass, R.T.: Temporal databases. In: Frank, A.U., Campari, I., Formentini, U. (eds.) GIS 1992. LNCS, vol. 639, pp. 22–64. Springer, Heidelberg (1992). https://doi.org/10.1016/S1574-6526(05)80016-1
14. Gutiérrez, C., Hurtado, C.A., Mendelzon, A.O., Pérez, J.: Foundations of Semantic Web databases. J. Comput. Syst. Sci. **77**(3), 520–541 (2011)

15. Geerts, F., Mecca, G., Papotti, P., Santoro, D.: Cleaning data with Llunatic. VLDBJ (2019). https://doi.org/10.1007/s00778-019-00586-5
16. Coelho, F.: DataFiller - generate random data from database schema (2014). https://www.cri.ensmp.fr/people/coelho/datafiller.html
17. Kaufmann, M., Fischer, P.M., May, N., Tonder, A., Kossmann, D.: TPC-BiH: a benchmark for bitemporal databases. In: Nambiar, R., Poess, M. (eds.) TPCTC 2013. LNCS, vol. 8391, pp. 16–31. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04936-6_2
18. Ao, J., et al.: Temporal Enrichment and Querying of Ontology-Compliant Data (Technical report TR-2020-3). https://www.csc.ncsu.edu/research/tech/reports.php
19. Nielsen, J.: Usability Engineering. Morgan Kaufmann, Burlington (1993)

Data Analytics

Bing-CSF-IDF+: A Semantics-Driven Recommender System for News

Lies Hooft van Huijsduijnen[1], Thom Hoogmoed[1], Geertje Keulers[1], Edmar Langendoen[1], Sanne Langendoen[1], Tim Vos[1], Frederik Hogenboom[1], Flavius Frasincar[1], and Tarmo Robal[2](✉)

[1] Erasmus University Rotterdam, Burgemeester Oudlaan 50, 3062 PA Rotterdam, The Netherlands
lieshooft@gmail.com, misterthom@gmail.com, geertje-k@gmail.com, e_langendoen@gmail.com, sannelangendoen@gmail.com, timokt93@gmail.com, {fhogenboom,frasincar}@ese.eur.nl

[2] Tallinn University of Technology, Ehitajate tee 5, 19086 Tallinn, Estonia
tarmo.robal@ttu.ee

Abstract. This work proposes the Bing-CSF-IDF+ recommender – a content-based recommender that makes use of semantic relationships, and combines the best features of our earlier introduced Bing-SF-IDF+ and CF-IDF+ systems. First, we make use of concepts and concept relationships from a domain ontology. Next, Bing-CSF-IDF+ employs the synsets and synset relationships from a semantic lexicon that have not been previously captured by the domain ontology. Last, named entities and their frequencies as provided by Bing – not present in the semantic lexicon and domain ontology – are utilized. Our experiments show that Bing-CSF-IDF+ significantly outperforms Bing-SF-IDF+ and CF-IDF+ on F_1-scores and Kappa statistics based on a news data set.

1 Introduction

The introduction of the World Wide Web at the end of the 20th century has resulted in a widely accessible knowledge source with enormous growth potential. According to estimates, the digital world almost doubles every two years in size, emerging in the prodigious scale of 44 trillion gigabytes in 2020 [20]. Hence, the challenge today is not to increase the amount of data, but to retrieve patterns of interest in this data.

Applications that deal with these problems and help to structure the overload of data are recommender systems (RS) [17]. RS are employed in various fields (e.g., news items, movies, books, etc.) to distinguish certain data based upon user's preferences which are captured in so-called *user profiles*, e.g., by using domain models [18]. The focus in this research is on the recommendation of news items. The large amounts of data available on every single news event does not facilitate search for users to find the items of their interest. News Web sites often categorize news items, however, these are not ordered according to the individual needs of a user. Therefore, users can strongly benefit from RS.

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 143–153, 2020.
https://doi.org/10.1007/978-3-030-54623-6_13

There are three main types of RS that can be used for this: *collaborative* RS, which provide recommendations based on similarities between preferences of one user and preferences of others, *content-based* RS, which recommend items according to their content, and *hybrid RS*, which are combinations of the former two approaches [3]. Here, our focus will be on content-based RS for news recommendation, as these enable a better understanding of the news item content, and are able to deal with the cold-item problems. We do not consider hybrid RS as we assume not to have a lot of information on users and their preferences.

The difficulty that arises in content-based recommendation is that machines are not able to understand the meaning of the text. This is, however, a necessary condition in order to provide suitable recommendations for the interests of a particular user. Therefore the words in the text need to be semantically analyzed, and the correct sense for each word determined by *word sense disambiguation*, enabled by using a semantic lexicon, e.g. WordNet [8]. Existing approaches such as (Bing-)CF-IDF+ [2,7] and (Bing-)SF-IDF+ [6,15] have used only subsets of these features, i.e., concepts and their relations for CF-IDF+, and synsets and their relations for SF-IDF+, and named entities for the Bing variant.

In this paper, we aim to make use of a larger set of features by combining those of the previously mentioned RS, and propose the Bing-CSF-IDF+ recommender. This content-based approach incorporates both the concepts found in the news item and their domain-specific related concepts, as well as the identified synsets and their related synsets using semantic relations from WordNet for mapping a user's preference. Moreover, named entities that are not present in WordNet or a domain ontology are considered. To find (related) concepts, a domain-specific ontology is used as knowledge base. We hypothesize that the proposed method, which combines different features of state-of-the-art recommendation methods, yields an improvement in news recommendation compared to existing RS. The performance of the Bing-CSF-IDF+ recommender will be measured by means of statistics, e.g., the F_1-measure and Kappa statistic.

The remainder of this paper is organized as follows. Section 2 discusses related work on content-based RS. Sections 3 and 4 provide a description of the proposed recommender, and present an evaluation against other recommenders as benchmark, respectively Section 5 discusses conclusions drawn from the conducted research and provides some directions for future work.

2 Related Work

Let us start with an overview of existing content-based RS, and consider traditional Vector Space Models (VSM) TF-IDF, CF-IDF, and SF-IDF, where TF-IDF is the oldest recommendation approach. The TF-IDF method is of interest as SF-IDF and CF-IDF build on the mathematical concept provided by TF-IDF. The Term Frequency - Inverse Document Frequency (TF-IDF) [19] recommender consists of two parts. The term frequency indicates how often a term occurs in a given news item. Higher frequencies are linked to higher relevancies. The inverse document frequency captures the importance of a term in a set of news items.

Frequent terms are considered to be common and less important. TF-IDF represents news items as term vectors containing scores, which can be compared to user vectors (aggregation of vectors corresponding to items previously consumed by the user) using similarity functions (e.g., cosine similarity). The TF-IDF score is large for terms that occur frequently in a particular news item but not often in all other news items. A certain specified threshold decides whether a news item and the user's interest are considered similar. The Synset Frequency - Inverse Document Frequency (SF-IDF) [5] VSM is a variation of TF-IDF, which in addition to all terms looks at synonyms and ambiguous terms using a semantic lexicon (WordNet). Terms having the same meaning will be subsumed in one single concept, and therefore, word sense disambiguation is needed. For terms with multiple meanings, corresponding word senses are to be counted separately. The Concept Frequency - Inverse Document Frequency (CF-IDF) [10] recommendation approach is another variant of TF-IDF, deviating from SF-IDF by using key ontological concepts instead of all synsets in a news item. CF-IDF considers news items as a weighted vector of concepts. A domain ontology linked to WordNet captures the most salient concepts of a domain.

The Semantic Relationship Vector Space Models extend the traditional VSM by taking semantic relationships into account. SF-IDF+ [15] extends the SF-IDF [5] method, by combining synsets with their synsets related using semantic relationships, such as WordNet hypernyms. The vector representation is extended by adding the related synsets from the synset of a news item to the vector representation, enabling better vector representation of news items. Bing-SF-IDF+ [6] is an extension of SF-IDF+, which in addition to words in the semantic lexicon also considers the similarity between named entities frequently occurring on the Web. The Bing similarity is based on the number of page counts originating from the Bing search engine. Each news item has an SF-IDF+ similarity value and a Bing similarity value with a user profile. A weighted average is used to compute the Bing-SF-IDF+ similarity value for a news item with a user profile, using the Point-Wise Mutual Information (PMI) [1] measure. The CF-IDF+ [7] recommender is an extension of CF-IDF [10], which also processes the news items into a concept vector representation but extends the model by considering related ontology concepts – direct super- and subclasses, and domain-specific related concepts, and their relationships. Only the related concepts not yet in the vector representation are taken into account, or if the related concept has a higher CF-IDF+ value than the previous value.

Lastly, for historical reasons we discuss two semantic similarity RS: SS and Bing-SS as both of these have been outperformed by Bing-SF-IDF+ [6] and CF-IDF+ [7]. The Semantic Similarity (SS) recommender [5] filters all possible user profile and news item synset pairs for words that do not have the same part-of-speech. Similarity scores are computed for the remaining pairs using various measures, e.g., Jiang and Conrath [12], Leacock and Chodorow [13], Lin [14], Resnik [16], and Wu and Palmer [21]. These similarity measures capture the distance between two synsets in a semantic graph (e.g., WordNet). Finally, the score for the unread news items is found by taking the average over the similarity scores for all pairs of synsets. Bing-SS [4] is an extension of the SS recommender. Similar to Bing-SF-IDF+, it additionally takes into account named entities in its computations for those synset pairs that have the highest similarity scores.

3 Bing-CSF-IDF+

The Bing-CSF-IDF+ recommender combines information from found named entities, concepts and their relationships, and synsets and their relationships by using the Bing, the CF-IDF+, and the SF-IDF+ similarity values for news items. As the previous recommenders, it also relies on the Hermes framework.

Hermes is a framework for indexing, querying, and recommending news items using a knowledge base [9]. It allows to construct the knowledge base from RSS (Really Simple Syndication) feeds, advantaging from the meta-data, e.g., the title, category, and publication date of the news items available in these feeds, and enables collecting news items from multiple news sources. Hermes also stores the user profile (created by collecting the concepts of interest from previously read news) containing information about the news items and subjects a user finds interesting. Last, Hermes uses a domain ontology (created by domain experts and defines relationships between different concepts pertinent to a certain domain), which enables semantic-based news indexing and querying. Using the content and the meta-data of the news items in the knowledge base (the instance) and the domain ontology (the schema), the news items are pre-processed into vector space models before being run through the recommenders. The implementation of the Hermes uses a Natural Language Processing (NLP) engine to pre-process the news articles and employs linguistic techniques such as lemmatization, word sense disambiguation, tokenization, sentence-splitting, and concept detection to find which concepts are described in the text. The latest description of the Hermes framework and recommender implementations can be found in [2].

The Bing-CSF-IDF+ recommender assumes a certain order of steps taken. First, a news item is analyzed on the presence of words which trigger concepts from the ontology. The words found to trigger concepts from the ontology are no longer considered in the next step, which is looking for named entities by means of the Bing method. These found named entities will now no longer be considered for the last step, which analyzes the remainder of the news item by means of the SF-IDF+ recommender. We have considered this order of processing steps as we assume that the ontology, followed by named entities, and then synsets provide for the most specific, and thus the most useful information when analyzing news.

The Bing-CSF-IDF+ similarity measure is calculated by linearly combining the weighted averages of the similarity values between a user profile and an unread news item found for the CF-IDF+ recommender, the Bing method, and the SF-IDF+ recommender:

$$\begin{aligned} \text{sim}_{\text{Bing-CSF-IDF+}}(d_u, d_r) = {} & \alpha \times \text{sim}_{\text{Bing}}(d_u, d_r) \\ & + \beta \times \text{sim}_{\text{CF-IDF+}}(d_u, d_r) \\ & + (1 - \alpha - \beta) \times \text{sim}_{\text{SF-IDF+}}(d_u, d_r), \end{aligned} \tag{1}$$

where d_r is the vector representation of the user's interest, d_u the vector representation of unread news items, and α, β predefined values (like with Bing-SF-IDF+ [6]), which can be optimized by means of a genetic algorithm to obtain the best performance of the Bing-CSF-IDF+ recommender on a validation data set. As with existing RS, unread news items for which the normalized similarity measure exceeds a predefined cut-off value are recommended.

4 Evaluation

The evaluation of the newly proposed recommender will be discussed through the set-up of the experiment, the optimization of the weights used in the recommender, and the results obtained for Bing-CSF-IDF+ and the existing RS.

4.1 Setup

The evaluation setup is similar to setups in existing literature on news recommendation [6,7]. The used data set consists of 100 different news articles, originating from a Reuters RSS feed. All these articles are concerned with financial news on technology companies. Next, 8 different user profiles are contained in this data set. Each user profile is linked to a specific topic. The topics are: "Asia", "Financial markets", "Google and its competitors", "Internet of Web services", "Microsoft and its competitors", "National economies", "Technology", and "United States". The user profiles were created by 3 researchers (experts in news analytics) from the Erasmus University Rotterdam by rating a news article as either interesting or not for a certain profile. Articles are considered to be interesting based on the principle of majority voting. Table 1 reports the user profile topics with their inter-annotator agreements (IAA), and the number of interesting (I+) and non-interesting (I−) news items as given by the experts.

Table 1. Number of interesting (I+) and non-interesting (I−) news items, and the inter-annotator agreement (IAA)

Topic	I+	I−	IAA
Asia or its countries	21	79	99%
Financial markets	24	76	72%
Google or its rivals	26	74	97%
Web services	26	74	94%
Microsoft or its rivals	29	71	98%
National economies	33	67	90%
Technology	29	71	87%
United States	45	55	85%

To test the performance of the recommenders, the complete data set of 100 news articles is split into smaller subsets – a training set (30%), a validation set (30%), and a test set (40%), keeping the proportions of relevant/non-relevant news items for each of the considered sets (we have 8 such divisions, one for each user profile). The training set is used for learning the user profile. The validation set is used for finding the optimal weights used in the recommender.

The recommender uses the knowledge about the user learned from the training and validation set to predict whether an article from the test set is interesting or not. An unread article is marked as interesting if the similarity value between the user profile and the article is higher than a predefined cut-off value. The news items classified by RS as interesting for the user, will be recommended.

4.2 Optimizing Weights

The optimization of the weights for the recommender is done using a Genetic Algorithm. The Genetic Algorithm works with sets of solutions, called populations. In each iteration of the algorithm, the previous population is adapted such that one navigates through the parameter space to the (local) optimum. For the Bing-CSF-IDF+ recommender there are 32 weights which need to be optimized, namely weights for 27 SF-IDF+ relations, 3 CF-IDF+ relations, and α and β. The optimization is done on the Lisa system from SURFsara[1]. The Lisa system is a computer cluster consisting of several hundreds of multi-core nodes and is meant for researchers who need large computing capacities. As the nodes of the Lisa system contain multiple cores, each computer can run multiple jobs in parallel. We want to find the optimal weights for several cut-off values. The optimization for different cut-off values does not depend on each other, so these jobs can be independently parallelized.

Note, as the genetic algorithm is a heuristic algorithm, the algorithm might not be able to find the optimal weights, but we were able to search large parameter spaces due to the fact that we had the computing power of the Lisa system at our disposal. This makes it likely that the optimized weights are nearly optimal.

4.3 Results

The following results for the considered recommenders were obtained on exactly the same splits of the data set. Note that these splits are different from the ones considered in the previous works [6,7].

First, the results for the F_1-measure are presented for the Bing-CSF-IDF+, Bing-SF-IDF+, and CF-IDF+ recommenders. These are the recommenders that are the most interesting for this research, as previous research showed that Bing-SF-IDF+ and CF-IDF+ recommenders gave the best results. Figure 1 outlines that the CF-IDF+ recommender seems to perform well for low cut-off values (i.e., in situations where low precision is tolerated in favor of high recall). From a cut-off value of about 0.3, both the Bing-CSF-IDF+ and Bing-SF-IDF+

[1] SURFsara: The Lisa System, https://userinfo.surfsara.nl/systems/lisa.

recommender perform notably better than the CF-IDF+ recommender. This is an indication that Bing-CSF-IDF+ and Bing-SF-IDF+ boast a high precision and are able to pull a higher recall in more strict recommendation contexts. The most important result that can be deduced from Fig. 1 is that the Bing-CSF-IDF+ recommender seems to perform at least as well as the other two recommenders for almost all cut-off values. Especially for the cut-off values which range from 0.05 to 0.4, the combination of both the CF-IDF+ recommender and the Bing-SF-IDF+ recommender for the Bing-CSF-IDF+ recommender, seems to be useful.

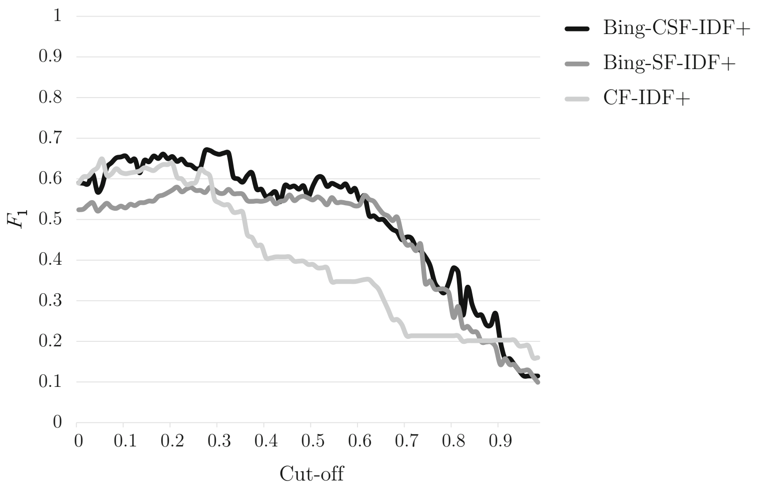


Fig. 1. F_1-measures for several recommenders

The observations made from Fig. 1 are confirmed by the Student-t test which was used to determine whether one recommender was statistically better than the other recommenders. The Student-t test was used for testing whether two recommenders had significantly different average F_1-measures. The p-values of this test can be found in Table 2. The performance of the recommenders from worst to best is CF-IDF+, Bing-SF-IDF+, and then Bing-CSF-IDF+. All results were found to be significant on a 5% significance level.

Also the Cohen's Kappa statistic was determined for each of the cut-off values and each of the recommenders. The Cohen's Kappa statistic measures the inter-rater agreement between the classifications and the actual interestingness of news articles (by taking into account the agreement by chance). Figure 2 shows the results for the Cohen's Kappa statistic. The Bing-SF-IDF+ recommender seems to perform notably worse than the other two recommenders for low cut-off values. The Bing-SF-IDF+ recommender, however, improves in relative performance for larger cut-off values. Again, the Bing-CSF-IDF+ recommender seems to perform at least as well as the other recommenders for almost all cut-off values.

Table 2. One-tailed two-sample Student-t test p-values for the F_1-measure (H_0: $\mu_{\text{column}} = \mu_{\text{row}}$, H_1: $\mu_{\text{column}} > \mu_{\text{row}}$, $\alpha = 0.05$)

	CF-IDF+	Bing-SF-IDF+	Bing-CSF-IDF+
CF-IDF+	–	0.00	0.00
Bing-SF-IDF+	1.00	–	0.00
Bing-CSF-IDF+	1.00	1.00	–

Once again, the observations made from Fig. 2 are confirmed by statistical tests. A Student-t test was performed on the average Kappa statistic for each of the recommenders (Table 3). This time it was found that the recommenders could be ordered from worst to best in the order of Bing-SF-IDF+, CF-IDF+, and Bing-CSF-IDF+. Again, all results are significant on a 5% significance level. So from both the F_1-measure and the Kappa statistic, we can conclude that the Bing-CSF-IDF+ recommender performs the best.

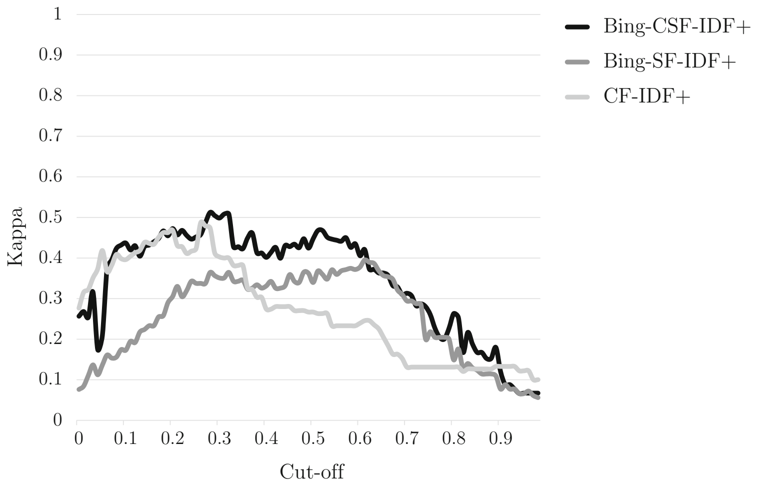


Fig. 2. Kappa statistics for several recommenders

The optimized weights for the different recommenders, while considering the optimal cut-off (with respect to F_1) for each recommender, have notable differences. For CF-IDF+, for the cut-off value of 0.06, each relationship is almost equally informative. For the cut-off value of 0.22, the Bing-SF-IDF+ α parameter has an optimized weight of 0.14220, indicating that the SF-IDF+ recommender contributes the most information. Last, for Bing-CSF-IDF+, α and β are optimized to 0.20268 and 0.50435, respectively, at a cut-off value of 0.28. Thus,

Table 3. One-tailed two-sample Student-t test p-values for the Kappa statistic (H_0: $\mu_{\text{column}} = \mu_{\text{row}}$, H_1: $\mu_{\text{column}} > \mu_{\text{row}}$, $\alpha = 0.05$)

	CF-IDF+	Bing-SF-IDF+	Bing-CSF-IDF+
CF-IDF+	–	1.00	0.00
Bing-SF-IDF+	0.00	–	0.00
Bing-CSF-IDF+	1.00	1.00	–

CF-IDF+ contributes about half the information, Bing contributes about 20% of the information and SF-IDF+ contributes the rest of the remaining 30%.

Looking at Fig. 1, the resulting weights for the Bing-CSF-IDF+ recommender make sense. For the cut-off value of 0.28 the CF-IDF+ recommender clearly performs better than the Bing-SF-IDF+, hence why the assigned weight is larger.

5 Conclusion

We have proposed a new semantics-driven Bing-CSF-IDF+ recommender combining the best features of the existing CF-IDF+ and Bing-SF-IDF+ recommenders. We have shown that the newly proposed Bing-CSF-IDF+ recommender outperforms the already existing recommenders. For almost each cut off value, both the F_1-measure and the Kappa statistic of the Bing-CSF-IDF+ recommender are at least as high as the other recommenders, meeting our expectations, as the Bing-CSF-IDF+ recommender could be transformed to both a pure CF-IDF+ as well as a Bing-SF-IDF+ recommender by choosing the appropriate values for the weights α and β. The occasions that the values of these statistics are higher for the existing recommenders can be explained by the fact that the genetic algorithm is used for optimizing all the weights simultaneously. As this algorithm is only a heuristic, it does not need to find the optimal weights.

We envision various possible opportunities and directions for future work. It would be interesting to compare the proposed method to graph embedding based recommendation [11]. Also, one could look for better heuristic algorithms to optimize the weights for the Bing-CSF-IDF+ recommender, for example, an Ant Colony Optimization algorithm. This algorithm might find better weights, which will improve the performance of the recommender.

Another improvement might be made by only taking the most similar named entities into account (as performed in the Bing-SS recommender), or by using machine learning algorithms (e.g., SVMs) to learn the recommender model using all the available features. Moreover, using WordNet relations also for concepts (if concepts have a synset associated) could possibly lead to a better performance of the recommender. Finally, the results could be improved by using a more extensive domain ontology for the CF-IDF+ values, to discover additional important concepts in the news articles that might influence the classification of an article as interesting or not.

References

1. Bouma, G.: Normalized (pointwise) mutual information in collocation extraction. In: Biennial GSCL Conference 2009 (GSCL 2009), pp. 31–40. Gunter Narr Verlag Tübingen (2009)
2. Brocken, E., et al.: Bing-CF-IDF+: a semantics-driven news recommender system. In: Giorgini, P., Weber, B. (eds.) CAiSE 2019. LNCS, vol. 11483, pp. 32–47. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21290-2_3
3. Burke, R.: Hybrid Recommender Systems: Survey and Experiments. User Model. User-Adap. Inter **12**(4), 331–370 (2002). https://doi.org/10.1023/A:1021240730564
4. Capelle, M., Hogenboom, F., Hogenboom, A., Frasincar, F.: Semantic news recommendation using wordnet and bing similarities. In: 28th Symposium on Applied Computing (SAC 2013), The Semantic Web and its Application Track, pp. 296–302. ACM (2013)
5. Capelle, M., Moerland, M., Frasincar, F., Hogenboom, F.: Semantics-based news recommendation. In: 2nd International Conference on Web Intelligence, Mining and Semantics (WIMS 2012). ACM (2012)
6. Capelle, M., Moerland, M., Hogenboom, F., Frasincar, F., Vandic, D.: Bing-SF-IDF+: a hybrid semantics-driven news recommender. In: 30th Symposium on Applied Computing (SAC 2015), Web Technologies Track, pp. 732–739. ACM (2015)
7. de Koning, E., Hogenboom, F., Frasincar, F.: News recommendation with CF-IDF+. In: Krogstie, J., Reijers, H.A. (eds.) CAiSE 2018. LNCS, vol. 10816, pp. 170–184. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91563-0_11
8. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
9. Frasincar, F., Borsje, J., Hogenboom, F.: Personalizing news services using semantic web technologies. In: Lee, I. (ed.) E-Business Applications for Product Development and Competitive Growth: Emerging Technologies, Chap. 13, pp. 261–289. IGI Global, Pennsylvania (2011)
10. Goossen, F., IJntema, W., Frasincar, F., Hogenboom, F., Kaymak, U.: News personalization using the CF-IDF semantic recommender. In: International Conference on Web Intelligence, Mining and Semantics (WIMS 2011). ACM (2011)
11. Grad-Gyenge, L., Kiss, A., Filzmoser, P.: Graph embedding based recommendation techniques on the knowledge graph. In: Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization (UMAP 2017), pp. 354–359. ACM (2017)
12. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. In: 10th International Conference on Research in Computational Linguistics (ROCLING 1997), pp. 19–33. ACLCLP (1997)
13. Leacock, C., Chodorow, M.: WordNet: An Electronic Lexical Database, Chap. Combining Local Context and WordNet Similarity for Word Sense Identification, pp. 265–283. MIT Press, Cambridge (1998)
14. Lin, D.: An information-theoretic definition of similarity. In: 15th International Conference on Machine Learning (ICML 1998), pp. 296–304. Morgan Kaufmann, Burlington (1998)
15. Moerland, M., Hogenboom, F., Capelle, M., Frasincar, F.: Semantics-based news recommendation with SF-IDF+. In: 3rd International Conference on Web Intelligence, Mining and Semantics (WIMS 2013). ACM (2013)

16. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: 14th International Joint Conference on Artificial Intelligence (IJCAI 1995), pp. 448–453. Morgan Kaufmann, Burlington (1995)
17. Ricci, F., Rokach, L., Shapira, B.: Recommender systems: introduction and challenges. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 1–34. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_1
18. Robal, T., Haav, H.-M., Kalja, A.: Making web users' domain models explicit by applying ontologies. In: Hainaut, J.-L., et al. (eds.) ER 2007. LNCS, vol. 4802, pp. 170–179. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76292-8_20
19. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5), 513–523 (1988)
20. Turner, V., Gantz, J.F., Reinsel, D., Minton, S.: The digital universe of opportunities: rich data and the increasing value of the internet of things. International Data Corporation, White Paper, IDC_1672 (2014)
21. Wu, Z., Palmer, M.S.: Verb semantics and lexical selection. In: 32nd Annual Meeting of the Association for Computational Linguistics (ACL 1994), pp. 133–138. ACL (1994)

QuRVe: Query Refinement for View Recommendation in Visual Data Exploration

Humaira Ehsan[1](✉), Mohamed A. Sharaf[2], and Gianluca Demartini[1]

[1] The University of Queensland, Brisbane, QLD, Australia
humairaehsan@gmail.com
[2] United Arab Emirates University, Al Ain, Abu Dhabi, UAE
msharaf@uaeu.ac.ae

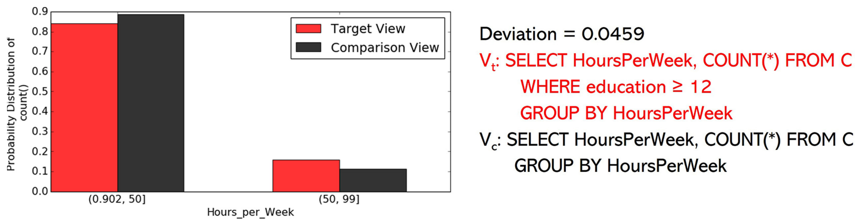
Abstract. The need for efficient and effective data exploration has resulted in several solutions that automatically recommend interesting visualizations. The main idea underlying those solutions is to automatically generate all possible views of data, and recommend the top-k interesting views. However, those solutions assume that the analyst is able to formulate a well-defined query that selects a subset of data, which contains insights. Meanwhile, in reality, it is typically a challenging task to pose an exploratory query, which can immediately reveal some insights. To address that challenge, this paper proposes to automatically refine the analyst's input query to discover such valuable insights. However, a naive query refinement, in addition to generating a prohibitively large search space, also raises other problems such as deviating from the user's preference and recommending statistically insignificant views. In this paper, we address those problems and propose the novel QuRVe scheme, which efficiently navigates the refined queries search space to recommend the top-k insights that meet all of the analysts's pre-specified criteria.

1 Introduction

Visual data exploration is the rudiment of deriving insights from large datsets. Typically, it involves an analyst performing the following steps: 1) selecting a subset of data, 2) generating different visualizations of that subset of data, and 3) sifting through those visualizations for the ones which reveal interesting insights. Based on the outcome of the last step, the analyst might have to refine their initial selection of data so that the new subset would show more interesting insights. This is clearly an iterative and time-consuming process, in which each selection of data (i.e., exploratory input query) is a springboard to the next one.

Motivated by the need for an efficient and effective visual data exploration process, several solutions have been proposed towards automatically finding and recommending interesting data visualizations (i.e., steps 2 and 3 above) (e.g., [6–8,14,17]). The main idea underlying those solutions is to automatically generate all possible views of the explored data, and recommend the top-k *interesting*

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 154–165, 2020.
https://doi.org/10.1007/978-3-030-54623-6_14



```
Deviation = 0.0459
Vt: SELECT HoursPerWeek, COUNT(*) FROM C
      WHERE education ≥ 12
      GROUP BY HoursPerWeek
Vc: SELECT HoursPerWeek, COUNT(*) FROM C
      GROUP BY HoursPerWeek
```

Fig. 1. View on input query Q

views, where the interestingness of a view is quantified according to some *utility* function. Recent work provides strong evidence that a deviation-based formulation of utility is able to provide analysts with interesting visualizations that highlight some of the particular trends of the analyzed datasets [8,14,17,19]. In particular, the deviation-based metric measures the distance between the probability distribution of a visualization over the analyzed dataset (i.e., *target view*) and that same visualization when generated from a comparison dataset (i.e., *comparison view*), where the comparison dataset is typically the entire database. The underlying premise is that a visualizations that results in a higher deviation is expected to reveal insights that are very particular to the analyzed dataset.

Existing solutions have been shown to be effective in recommending interesting views under the assumption that the analyst is "precise" in selecting their analyzed data. That is, the analyst is able to formulate a well-defined exploratory query, which selects a subset of data that contains interesting insights to be revealed by the recommended visualizations. However, such assumption is clearly impractical and extremely limits the applicability of those solutions. In reality, it is typically a challenging task for an analyst to select a subset of data that has the potential of revealing interesting insights. Hence, it is a continuous process of trial and error, in which the analyst keeps *refining* their selection of data manually and iteratively until some interesting insights are revealed. Therefore, in this work we argue that, in addition to the existing solutions for automatically recommending interesting views, there is an equal need for solutions that can also automatically select subsets of data that would potentially provide such interesting views. Hence, our goal in this work is not only to recommend interesting views, but also to recommend exploratory queries that lead to such views. To further illustrate the need for such solution, consider the following example.

Example 1. Consider an analyst wants to explore and find interesting insights in the U.S. Census income dataset [1], which is stored in table C. Her intuition is that analyzing the subset of data of those who have achieved a high level of education might reveal some interesting insights. Therefore, she selects that particular subset in which everyone has completed their 12th year of education (i.e., graduated high school) via the query: Q: `SELECT * FROM C WHERE education` $\geq$ `12`. To find the top-k visualizations, she might use one of the existing approaches (e.g., [8,17]), in which all the target and comparison aggregate views are generated and their deviation is computed by using a distance function (e.g., Euclidean distance). Figure 1 shows the top-k visualization recommended by such approaches. Particularly, the figure shows a bar chart in which the x-axis is the dimension `Hours per week`, and the y-axis is the probability distribution of the aggregate function `COUNT`. Such visualization is equivalent to plotting the probability distributions of the target view V_t and the comparison View V_c, which are expressed in SQL in Fig. 1. Hence, the deviation value shown in Fig. 1 is the Euclidean distance between the probability distribution of V_t and V_c. However, Fig. 1 clearly shows that the target and comparison views are almost the same, which is also reflected by the low-deviation value of 0.0459. However, such visualization would still be recommended by existing approaches because it achieves the maximum deviation among all the views generated over the data subset selected by query Q.

The previous example illustrates a clear need for a query refinement solution that is able to automatically modify the analyst's initial input query and recommend a new query, which selects a subset of data that includes interesting insights. To that end, one straightforward and simple approach would involve generating all the possible subsets of data by automatically refining the predicates of the input query. Consequently, for each subset of data selected by each query refinement, generate all possible aggregate views (i.e., visualizations). In addition to the obvious challenge of a prohibitively large search space of query refinements, that naive approach would also lead to visualizations that might appear to be visually interesting but they are irrelevant from the analyst's perspective. Particularly, there are two issues with that approach, and in turn the recommended visualization; 1) *similarity-oblivious*: a blind automated refinement that is oblivious to the analyst's preferences might result in a refined query that is significantly dissimilar from the input query, and 2) *statistical insignificance*: the subset selected by the refined query can be too small and as a result the target views generated from that subset will miss a number of values for the dimension attribute. This leads to views with high deviation values but statistically insignificant.

The two issues mentioned above highlight the need for automatic refinement solutions that are guided by the user's preference and statistical significance, which is the focus of this work. In particular, we propose a novel scheme **QuRVe**, which is particularly optimized to leverage the specific features of the problem for pruning that large search space, as explained in the next sections.

2 Preliminaries

2.1 View Recommendation

Similar to the recent data visualization platforms [8,17], we are given a multi-dimensional dataset $D(\mathbb{A}, \mathbb{M})$, where $\mathbb{A}$ is the set of dimension attributes, $\mathbb{M}$ is the set of measure attributes, and $\mathbb{F}$ is the set of possible aggregate functions over the measure attributes $\mathbb{M}$. In a typical visual data exploration session the user chooses a subset D_S of the dataset D by issuing an input query Q. For instance, consider the query Q: `SELECT * FROM` D `WHERE T;` . In Q, T specifies a combination of predicates, which selects D_S for visual analysis (e.g., `education` $\geq$ `12` in Ex. 1). A visual representation of Q is basically the process of generating an aggregate view V_i of its result (i.e., D_S), which is then plotted using some visualization methods such as bar charts, scatter plots, etc. Therefore, an aggregate view V_i over D_S is represented by a tuple (A, M, F, b) where $A \in \mathbb{A}$, $M \in \mathbb{M}$, $F \in \mathbb{F}$ and b is the number of bins in case A is numeric. That is, D_S is grouped by dimension attribute A and aggregated by function F on measure attribute M. For instance, the tuple `(Hours per Week, *,COUNT,2)` represents the aggregate view shown in Fig. 1.

Towards automated visual data exploration, recent approaches have been proposed for recommending interesting visualizations based on deviation based metric (e.g., [8,17]). In particular, it measures the deviation between the aggregate view V_i generated from the subset data D_S vs. that generated from the entire database D, where $V_i(D_S)$ is denoted as *target* view, whereas $V_i(D)$ is denoted as *comparison* view. To ensure that all views have the same scale, each target view $V_i(D_S)$ and comparison view $V_i(D)$ is normalized into a *probability distribution* $P[V_i(D_S)]$ and $P[V_i(D)]$ and it is bounded by the maximum deviation value D_M. Accordingly, the deviation $D(V_i)$, provided by a view V_i, is defined as the normalized distance between those two probability distributions.

$$D(V_i) = \frac{dist(P[V_i(D_S)], P[V_i(D)])}{D_M} \tag{1}$$

Then, the deviation $D(V_i)$ of each possible view V_i is computed, and the k views with the highest deviation are recommended (i.e., *top-k*) [8,9,17,19]. However, to ensure that those top-k recommended views reveal interesting insights, we propose utilizing query refinement techniques, which are explained next.

2.2 Query Refinement

Automatic query refinement is a widely used technique for DBMS testing, information retrieval and data exploration. In a nutshell, in this technique the user provides an initial query and then it is progressively refined to meet a particular objective [13,16,18,20]. In this work, we propose to automatically refine an input exploratory query for the objective of view recommendation. Particularly, as mentioned in Sect. 2.1, the user provides an input query Q, which is progressively refined by automatically enumerating all combinations of predicates for the objective of generating interesting views.

Particularly, we consider queries having selection predicates with range $(<, \leq, >, \geq)$ operators. These predicates are defined on a set of numeric dimension attributes denoted as $\mathbb{P}$. The number of predicates is p, such that $|\mathbb{P}| = p$. Each of this range predicate is in the form $l_i \leq P_i \leq u_i$ where $P_i \in \mathbb{P}$ and l_i and u_i are the lower and upper limits of query Q along predicate P_i. The domain of predicate P_i is limited by a Lower bound L_i and upper bound U_i. A refined query Q_j is generated by modifying the lower and/or upper limits for some of the predicates in Q. That is, for a predicate $l_i \leq P_i \leq u_i$ in query Q, a refined predicate in Q_j takes the form $l'_i \leq P_i \leq u'_i$. Similar to [2,13], we convert a range predicate into two single-sided predicates. Therefore, $l_i \leq P_i \leq u_i$ is converted to two predicates: $P_i \leq u_i \bigwedge -P_i \leq -l_i$. This allows refinement of one or both sides of the range predicates and this results in the total number of single sided predicates to be $2p$. The set of all of the refined queries is denoted as $\mathbb{Q}$.

A refined query Q_j is obtained by changing one or more predicates $P_i \in T$ to P'_i, which naturally makes the refined query Q_j different from the input query Q. However, a refined query that is significantly dissimilar from its counterpart input query would result in loss of user preference and might be deemed irrelevant to the analysis. Hence, to quantify the change made to transform Q into the refined query Q_j, we define a similarity measure $S(Q, Q_j)$ in terms of the distance between Q_j and Q (i.e., $s(Q, Q_j)$).

$$S(Q, Q_j) = 1 - s(Q, Q_j) \tag{2}$$

While the exact specification of $s(Q, Q_j)$ is deferred to Sect. 4, it is worth pointing out the impact of query refinement on the deviation computation defined in Eq. 1. Particularly, when utilizing refinement, a view V_i can be either generated from the input query, or a refined one. To associate each view with its underlying query, we denote a view as V_{i,Q_j} to specify the i^{th} view generated over the result of query Q_j. Accordingly, Eq. 1 is modified to define the deviation $D(V_{i,Q_j})$ of a view V_{i,Q_j}, as:

$$D(V_{i,Q_j}) = \frac{dist(P[V_i(D_{Q_j})], P[V_i(D)])}{D_M} \tag{3}$$

2.3 Hypothesis Testing

In visual data exploration, it is often the case that an observed high-deviation is actually statistically insignificant. This problem leads to misleading ranking of such views, and in turn inaccurate recommendations [3,4,21]. For instance, in our recent work on the *MuVE* scheme [8,9], we made the following observations:

1. Some of the recommended top-k target views have very few underlying tuples, which lead to higher deviation values. Consequently, such views receive higher rank despite of the lack of real insight.
2. Often the data selected by the exploratory query result in only low-deviation views. Consequently, the top-k recommend views will exhibit low-deviation, as shown in Ex. 1. However, such top-k recommendations are clearly statistically insignificant.

To determine whether the observed difference is statistically significant, we employ the widely used approach hypothesis testing. Hypothesis testing determines if there is enough evidence for inferring that a difference exists between two compared samples or between a sample and population. A difference is called statistically significant if it is unlikely to have occurred by chance [3]. Hypothesis testing involves testing a null hypothesis by comparing it with an alternate hypothesis. The hypothesis to be tested is called the *null hypothesis*, denoted as H_0. The null hypothesis states that there is no difference between the population and the sample data. The null hypothesis is tested against an *alternate hypothesis*, denoted as H_1, which is what we have observed in the sample data. For instance, in Fig. 1 of Ex. 1, the hypothesis is that *"high school graduates work different number of hours per week (Hours worked is divided into two categories) as compared to the population"*, and this becomes H_1. The corresponding H_0 is that no such difference exits. Likewise, each possible view V_i from each refined query become a H_1, which is to be tested for significance before recommendation.

Depending on the nature of the statistical test and the underlying hypothesis, different null hypothesis statistical tests have been developed, e.g., chi-square test for categorical dimension attributes. Furthermore, after stating H_0 and H_1, the chosen statistical test returns *p-value*. The p-value is the probability of obtaining a statistic at least as extreme as the one that was actually observed, given H_0 is true. Specifically, the p-value is compared against a priori chosen *significance level* α, where the conventionally used significance level is 0.05. Hence, if $pvalue(V_i) \leq \alpha$, then H_0 must be rejected, which means the V_i is statistically significant. Clearly, due to the nature of the statistical test involved, the acceptance or rejection of H_0 can never be free of error. If the test incorrectly rejects or accepts H_0, then an error has occurred. Hypothesis testing can incur the following two types of error: 1) If H_0 is rejected, while it was true, it is called *Type-I error* and 2) If H_0 is accepted, while H_1 was true, it is called *Type-II error*. Type-II error is critical in our case because we do not want to reject views that might be interesting. The probability of Type-II error is specified by a parameter β, which normally has a value $0.10 - 0.20$. An alternate term is power, which is the probability of rejecting a false H_0, therefore, $power = 1 - \beta$. A priori power analysis is employed to determine the minimum sample size that is necessary to obtain the required power. By setting an effect size (ω), significance level (α), and power level (β), the sample size to meet specification can be determined [5].

3 Query Refinement for View Recommendation

In a nutshell, the goal of this work is to recommend the top-k bar chart visualizations of the results of query Q and all its corresponding refined queries $Q_j \in \mathbb{Q}$, according to some utility function. However, that simple notion of utility falls short in capturing the impact of refinement on the input query. In particular, automatic refinement introduces additional factors that impact the level of interestingness, and in turn utility of the recommended views. Accordingly, in our proposed scheme, we employ a weighted multi-objective utility function and constraints to integrate such factors. In particular, for each view V_{i,Q_j}, we evaluate the following components:

1. *Interestingness:* Is the ability of view V_{i,Q_j} to reveal some insights about the data, which is measured using the deviation-based metric $D(V_{i,Q_j})$ (Eq. 3).

2. *Similarity:* Is the similarity between the input query Q, and the refined query Q_j underlying the view V_{i,Q_j}, which is measured as $S(Q, Q_j)$ (Eq. 2).
3. *Statistical Significance:* Is the ability of the refined query Q_j and the view V_{i,Q_j} to generate a statistically significant result, which is captured by checking that the size of the subset selected by Q_j satisfies the constraint $power(Q_j)$, and the significance of the view V_{i,Q_j} satisfies the constraint $pvalue(V_{i,Q_j})$.

To capture the factors and constraints mentioned above, we employ a weighted multi-objective utility function, which is defined as follows:

$$U(V_{i,Q_j}) = \alpha_S \times S(Q, Q_j) + \alpha_D \times D(V_{i,Q_j}) \tag{4}$$

Parameters α_S and α_D specify the weights assigned to each objective in our hybrid utility function, such that $\alpha_S + \alpha_D = 1$. Those weights can be user-defined so that to reflect the user's preference between interestingness and similarity. Also, notice that all objectives are normalized in the range $[0, 1]$.

To fully define the similarity component of our utility function, we revisit Eq. 2 which quantifies the distance between Q and Q_j. In literature, a number of methods have been proposed to measure the distance between two range queries [11,15,16]. Similar to [2,18], we calculate the distance in terms of absolute change in predicate values (Eq. 5). This method provides a reasonable approximation of the change in data selected by the refined query at a negligible cost. Additionally, we normalize it by predicate bounds to accommodate the different scales of various predicates.

$$s(Q, Q_j) = \frac{1}{p} \sum_{i=1}^{p} \frac{|l_i^{Q_j} - l_i^{Q}| + |u_i^{Q_j} - u_i^{Q}|}{2|U_i - L_i|} \tag{5}$$

Definition: Query Refinement for View Recommendation: *G*iven a user-specified query Q on a database D, a multi-objective utility function U, a significance level α, statistical power $1 - \beta$ and a positive integer k. Find k aggregate views that have the highest utility values, from all of the refined queries $Q_j \in \mathbb{Q}$ such that $pvalue(V_{i,Q_j}) \leq \alpha$ and $power(Q_j) > 1 - \beta$.

In short, the premise is that a view is of high utility for the user, if it satisfies the specified constraints, shows high-deviation, and is based on a refined query that is highly similar to the user specified query.

4 Search Schemes

For an input query Q, each possible query refinement of Q can be represented as a point in p-dimensional space, where $|\mathbb{P}| = p$ (please see Sect. 2.2 for more details). Clearly, one of the points in that space is the input query Q itself, and the remaining points belong to the set of refined queries $\mathbb{Q}$. Our high-level goal is to: 1) generate the set $\mathbb{Q}$, 2) compute the utility of all the aggregate views generated from each query in $\mathbb{Q}$, and 3) recommend the top-k views after ranking them based on their achieved utility. To that end, clearly the large size of $\mathbb{Q}$ and the corresponding aggregate views, together with the complexity of evaluating the statistical significance and utility function of each view, makes the problem highly challenging. Hence, in this section, we put forward various search strategies for finding the top-k views for recommendation.

4.1 The Linear Scheme

Clearly, a naive way to identify the top-k objects is to score all objects based on a scoring function, sort, and return the top-k objects. Accordingly, the Linear scheme is basically an exhaustive and brute force strategy, in which views from all refined queries are generated and ranked according to their utility. As we consider predicates on continuous dimensions, infinite possible values can be assigned to predicates in those refined queries. Therefore, each dimension is discretized with a user specified parameter γ. This divides the range of dimension attribute into $1/\gamma$ equi-width intervals. In this scheme, irrespective of Q, iteratively all refined queries are generated using all combinations of Predicates $P_1, P_2 ... P_p$.

For each query $Q_j \in \mathbb{Q}$, to check the constraint $power(Q_j) < 1 - \beta$, a function *powerTest(*Q_j, ω, β, α*)* is defined, which returns true value if the constraint is satisfied, else it returns false. The cost of checking this constraint is one database probe, where a `COUNT` query with predicates of Q_j is executed to get the sample size of Q_j. Moreover, for the queries that satisfy the statistical power constraint, all views are generated. Then for each view V_{i,Q_j} the constraint $pValue(V_{i,Q_j}) < \alpha$ is checked. Specifically, for this purpose, another function *significanceTest(*V_{i,Q_j}, α*)* is defined, which returns a true value if p-value $< \alpha$. Consequently, for each view V_{i,Q_j} that satisfies the constraint, its utility value $U(V_{i,Q_j})$ is computed, and finally the top-k views are returned.

4.2 The QuRVe Scheme

Clearly, the linear search scheme, visits every possible view, therefore, it is very expensive in terms of execution time. In this section, we present the QuRVe scheme, which reduces cost by pruning a large number of views. Notice that our problem of finding top-k views is similar to the problem of top-k query processing, which is extensively studied in various settings [12]. Generally in these settings objects are evaluated by multiple objectives that contribute to the overall score of each object. In terms of efficiency, the best performing techniques for various top-k problem settings are based on the threshold algorithm (TA) [10,12]. TA generates sorted lists of objects on partial scores for every objective, visits the lists in round robin fashion and merges objects from different lists to compute the aggregated scores. Typically, it *early terminates* the search as soon as it has the *top-k* objects.

In our settings, we have a similar configuration i.e., we have two partial scores of a view V_{i,Q_j}, namely: 1) Similarity score $S(Q, Q_j)$, 2) Deviation score $D(V_{i,Q_j})$. These are stored in S_{list} and D_{list}. Conversely, we also have some key differences: 1) for any view V_{i,Q_j} the values of $S(Q, Q_j)$ and $D(V_{i,Q_j})$ are not physically stored and are computed on demand, 2) calculating $D(V_{i,Q_j})$ for a view is an expensive operation, and 3) the size of the view search space is prohibitively large and potentially infinite.

Obviously, a forthright implementation of TA is infeasible to our problem due to the limitations mentioned before. However, recall that the similarity objective $S(Q, Q_j)$ is the comparison of predicates of Q_j with Q and involves no database probes. Hence, a sorted list S_{list} can be easily generated at a negligible cost. However, populating the D_{list} in a similar fashion is not possible, as it involves expensive database probes. Therefore, to minimize the number of probes and efficiently populate D_{list}, the Sorted-Random (SR) model of the TA algorithm [12] is employed. In the SR model the sorted list (S) provides initial list of candidates and the random list (R) is probed only when required. Accordingly, QuRVe provides S_{list} as the initial list of candidate views, by incrementally generating refined queries in decreasing order of similarity and populating the S_{list}. The views in S_{list} have their partial scores, the final scores are only calculated for the views for which the D_{list} is also accessed. To achieve this, QuRVe maintains the variables U_{Unseen}: Stores the maximum utility of the views that are not probed yet and U_{Seen}: Stores the k^{th} highest utility of a view seen so far. Specifically, to calculate U_{Unseen}, the upper bound on deviation is used. Particularly, consider V_{i,Q_j} as the next view in S_{list} and let the upper bound on its deviation be $D_u(V_{i,Q_j})$. Moreover, let the upper bound on deviation from all views be D_u then $D_u = Max[D_u(V_{i,Q_j})]$. Consequently, $U_{Unseen} = \alpha_S \times S(Q, Q_j) + (1 - \alpha_S) \times D_u$. As the $D(V_{i,Q_j})$ is the normalized deviation, hence theoretically $D_u = Max[D_u(V_{i,Q_j})] = 1$.

In detail, QuRVe starts with initializations as: (i) there are no views generated yet, therefore $U_{Seen} = 0$, (ii) $U_{Unseen} = D_u$, and (iii) Q has the highest similarity i.e., $S(Q, Q) = 1$, therefore, Q is added to Q_{list} as the first member. Then, the power of the currently under consideration query Q_j is checked by the function *powerTest(*Q_j, ω, β, α*)*. Next, the corresponding views are generated by the function *generateViews(*Q_j*)* and the statistical significance test is performed on each view by the function *significanceTest(*V_{i,Q_j}, α*)*. The Utility of the views that pass the test is computed. Accordingly, the list *topk* is updated. The utility of k^{th} highest view is copied into U_{Seen}, to maintain the bound on the seen utility values. This completes processing the currently under consideration query Q_j. Later, the next set of neighboring queries are generated. In next iteration, another query Q_j is taken from the Q_{list} in order of the similarity objective value and accordingly the value of U_{Unseen} is updated. The iteration continue, until either there are no more queries to process, or the utility of the remaining queries will be less than the already seen utility (i.e., $U_{Unseen} > U_{Seen}$ is false). If QuRVe terminates because of the first condition that means its cost is the same as Linear search, as the optimization did not get a chance to step in. However, often QuRVe terminates because of the second condition (i.e., $U_{Unseen} > U_{Seen}$ is false) and achieves early termination. The most efficient performance of QuRVe is expected when U_{Unseen} decreases quickly during search and early termination can be triggered.

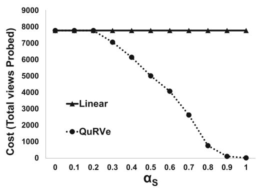


Fig. 2. Impact of α_S and α_D on cost

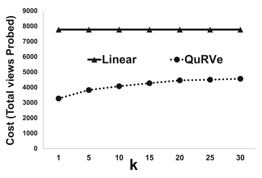


Fig. 3. Impact of k, $\alpha_S = 0.6$

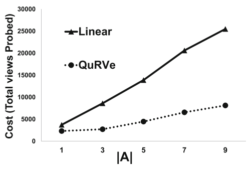


Fig. 4. Impact of dimensions

5 Experimental Evaluation

We perform extensive experimental evaluation to measure both the efficiency and effectiveness of out proposed schemes for top-k view recommendation.

Data Analysis: We assume a data exploration setting in which multi-dimensional datasets are analyzed. We use *CENSUS:* the census income dataset [1]. The dataset has 14 attributes and 48,842 tuples. The independent categorical attributes of the dataset are used as dimensions (e.g., occupation, work class, hours per week, sex, etc.), whereas the observation attributes are used as measures (capital gain, etc.) and the numerical independent attributes are used for predicates (e.g., education, age, etc.). The aggregate functions used are SUM, AVG and COUNT. In the analysis, all the α_S is in the range $[0 - 1]$, where $\alpha_S + \alpha_D = 1$. In default settings $\alpha_S = 0.5$, $k = 10$ and $\gamma = \frac{1}{2^3}$. For the purpose of statistical significance in experiments we use chi-square goodness of fit test.

Performance: We evaluate the efficiency and effectiveness of the different recommendations strategies in terms of the cost incurred. As mentioned in Sect. 3, the cost of a strategy is the total cost incurred in processing all the candidate views. We use the total views probed as the cost metric. Each experiment is performed with 10 randomly generated input queries, spread around the search space, then average of the cost is taken.

Impact of the α Parameters (Fig. 2): In this experiment, we measure the impact of the α values on cost. Figure 2 shows how the cost of *Linear* and *QuRVe* schemes are affected by changing the values of α_S. In Fig. 2, α_S is increasing while α_D is implicitly decreasing. Notice that the *Linear* scheme has the same cost for all values of α_S, which is as expected since it performs exhaustive search over all views. Therefore, its cost depends on the number of all combinations, irrespective of α. Figure 2 also shows that *QuRVe* has almost the same cost as *Linear* for $\alpha_S = 0$ and $\alpha_S = 0.1$, but outperforms it as the value of α_S increases. This is because in *QuRVe*, the upper bound on deviation is set to the theoretical maximum bound i.e., $D_u = 1$ and when $\alpha_S = 0$, $U_{Unseen} = 0 \times S + 1 \times D_u = 1$, consequently early termination is not possible. On the contrary, as α_S increases, chances of applying early termination based on the similarity value becomes possible. Hence, this prunes many database probes.

Impact of k (Fig. 3): Figure 3 shows that *Linear* is insensitive to the increase in the value of k, because it visits all views and sorts them according to their utility irrespective of the value of k. For $\alpha_S = 0.6$, *QuRVe* performs better than *Linear* for all values of k, due to the early termination feature of *QuRVe* scheme.

Scalability (Fig. 4): The search space of our problem depends on p, γ, $|A|$, $|M|$ and $|F|$. Increasing any of these factors explodes the search space. Consequently, the cost of all schemes increases as there are more views that are visited in search for the top-k views. Figure 4 shows the impact of number of the dimensions on cost. Particularly for this experiment the number of dimensions ($|A|$) are increased from 1 to 9. Figure 4 shows that the increase in the cost of *Linear* is linear with the increase in $|A|$. However, for *QuRVe* the increase in the cost is slow. Particularly, the cost of *QuRVe* increases as the number of dimension are increased from 1 to 3, because the schemes search from more views to generate top-k views. But for $|A| = 5$ the cost decreases this is because a new dimension is added which had views with high deviation and that resulted in increasing the value of U_{Seen} and triggering early termination.

6 Conclusions

Motivated by the need for view recommendation that lead to interesting discoveries and avoid common pitfalls such as random or false discoveries. In this paper we formulated the problem of query refinement for view recommendation and proposed QuRVe scheme. QuRVe refines the original query in search for more interesting views. It efficiently navigates the refined queries search space to maximize utility and reduce the overall cost. Our experimental results show employing the QuRVe scheme offer significant reduction in cost.

Acknowledgement. This work is partially supported by UAE University grant G00003352.

References

1. https://archive.ics.uci.edu/ml/datasets/adult
2. Albarrak, A., Sharaf, M.A., Zhou, X.: SAQR: an efficient scheme for similarity-aware query refinement. In: Bhowmick, S.S., Dyreson, C.E., Jensen, C.S., Lee, M.L., Muliantara, A., Thalheim, B. (eds.) DASFAA 2014. LNCS, vol. 8421, pp. 110–125. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05810-8_8
3. Bay, S.D., Pazzani, M.J.: Detecting group differences: mining contrast sets. Data Min. Knowl. Disc. **5**(3), 213–246 (2001). https://doi.org/10.1023/A:1011429418057
4. Chung, Y., et al.: Towards quantifying uncertainty in data analysis & exploration. IEEE Data Eng. Bull. **41**(3), 15–27 (2018)
5. Cohen, J.: Statistical Power Analysis for the Behavioral Sciences, revised edn. Academic Press, Cambridge (1977)
6. Demiralp, Ç., et al.: Foresight: recommending visual insights. PVLDB **10**(12), 1937–1940 (2017)

7. Ding, R., et al.: Quickinsights: quick and automatic discovery of insights from multi-dimensional data. In: SIGMOD, pp. 317–332 (2019)
8. Ehsan, H., et al.: MuVE: efficient multi-objective view recommendation for visual data exploration. In: ICDE, pp. 731–742 (2016)
9. Ehsan, H., et al.: Efficient recommendation of aggregate data visualizations. IEEE Trans. Knowl. Data Eng. **30**(2), 263–277 (2018)
10. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci. **66**(4), 614–656 (2003)
11. Kadlag, A., Wanjari, A.V., Freire, J., Haritsa, J.R.: Supporting exploratory queries in databases. In: Lee, Y.J., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 594–605. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24571-1_54
12. Marian, A., et al.: Evaluating top- k queries over web-accessible databases. ACM Trans. Database Syst. **29**(2), 319–362 (2004)
13. Mishra, C., Koudas, N.: Interactive query refinement. In: EDBT (2009)
14. Sellam, T., et al.: Ziggy: characterizing query results for data explorers. PVLDB **9**(13), 1473–1476 (2016)
15. Telang, A., et al.: One size does not fit all: toward user and query dependent ranking for web databases. IEEE Trans. Knowl. Data Eng. **24**(9), 1671–1685 (2012)
16. Tran, Q.T., et al.: How to conquer why-not questions. In: SIGMOD (2010)
17. Vartak, M., et al.: SEEDB: efficient data-driven visualization recommendations to support visual analytics. PVLDB **8**(13), 2182–2193 (2015)
18. Vartak, M., et al.: Refinement driven processing of aggregation constrained queries. In: EDBT (2016)
19. Wang, C., et al.: Efficient attribute recommendation with probabilistic guarantee. In: KDD, pp. 2387–2396 (2018)
20. Wu, E., et al.: Scorpion: explaining away outliers in aggregate queries. PVLDB **6**(8), 553–564 (2013)
21. Zhao, Z., et al.: Controlling false discoveries during interactive data exploration. In: SIGMOD (2017)

A Bloom Filter-Based Framework for Interactive Exploration of Large Scale Research Data

Gajendra Doniparthi[1(✉)], Timo Mühlhaus[2], and Stefan Deßloch[1]

[1] Heterogeneous Information Systems Group, University of Kaiserslautern, Kaiserslautern, Germany
doniparthi@informatik.uni-kl.de, muehlhaus@bio.uni-kl.de
[2] Computational Systems Biology, University of Kaiserslautern, Kaiserslautern, Germany
dessloch@informatik.uni-kl.de

Abstract. We present a novel RDBMS-based framework for interactively querying and exploring large-scale bio-science research data. We focus on the interactive exploration model and its evaluation support using Bloom filter indexing techniques for Boolean containment expressions. In particular, our framework helps explore structured research data augmented with schema-less contextual information. Our experiments show significant improvements over traditional indexing techniques, enabling scientists to move from batch-oriented to interactive exploration of research data.

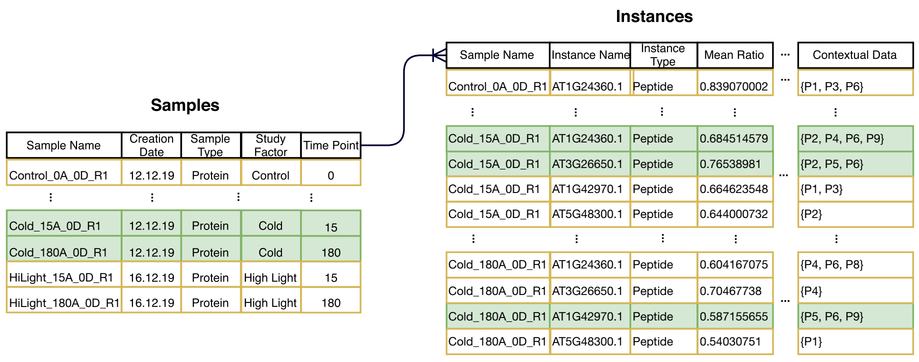
Keywords: Interactive data exploration · Relational JSON · Cross-omics · Bloom filter indices · Research data management

1 Introduction

With the advances in scientific research and big data processing, scientists now have access to large amounts of heterogeneous research data thus opening up opportunities for integrated exploration to create new and meaningful scientific knowledge. In particular, the standard frameworks for bio-science experiments follow hierarchical models [6] which makes them trivial to implement and maintain using conventional relational databases. However, augmented data such as instrument parameters, contextual information, etc., is schema-less and does not blend well with the n-ary storage model of the relational approach. The sparsity and the volume of the attribute-value format data make it difficult to design data management systems that allow analysts to interactively explore the research data at acceptable latencies [7]. Although traditional RDBMS systems offer data types such as JSON for schema-less development, the indexing support is still not extensive [5] and the typical one-shot database querying approach is not suitable for interactive exploration. In this paper, we introduce a vertically-scalable interactive exploration framework that uses a simple relational model plus JSON and a fast, scalable, and space-efficient Bloom filter-based indexing technique, specifically built for exploratory data analysis. We discard the typical one-shot database queries approach and instead allow data analysts to explore research data progressively while interactively evaluating Boolean containment filter conditions on huge volumes of contextual information, all in-memory. We designed our framework keeping in mind the typical research data management applications where *writes* and *reads* are frequent but *updates* and *deletes* are negligible.

1.1 Motivating Example

The data snippet in Fig. 1 shows acclimation reactions on a set of plant samples through varying temperature and light. In the study, the abundance of proteins from each sample is found by measuring the peptide instances under various experimental conditions. For simplicity, the contextual parameters and values per each instance are shown as attributes $P_1, P_2, ..., P_n$. Different Boolean combinations of these contextual parameters form the *Modification Patterns*, say, $((P_m \cup P_n) \cap P_k)$, $(P_m > 1.8 \ \cup P_k = high)$ etc. To find the list of *Samples* and corresponding *Instances* containing a given modification pattern requires relational join and the containment predicate condition evaluated on the JSON column. However, in typical bio-science data analysis, the modification patterns are not known in advance and they are constructed at the time of analysis. Moreover, the modification patterns are varied and applied on the same data set to extract multiple result sets which are fed to the subsequent computation workflows, thus, making the one-shot querying approach time consuming and impractical for large databases. Hence, in our framework, we leverage Bloom filters to memorize the query result for each attribute beforehand and let the analysts compose and evaluate the modification patterns interactively during analysis.



Samples

Sample Name	Creation Date	Sample Type	Study Factor	Time Point
Control_0A_0D_R1	12.12.19	Protein	Control	0
⋮	⋮	⋮	⋮	
Cold_15A_0D_R1	12.12.19	Protein	Cold	15
Cold_180A_0D_R1	12.12.19	Protein	Cold	180
HiLight_15A_0D_R1	16.12.19	Protein	High Light	15
HiLight_180A_0D_R1	16.12.19	Protein	High Light	180

Instances

Sample Name	Instance Name	Instance Type	Mean Ratio	...	Contextual Data
Control_0A_0D_R1	AT1G24360.1	Peptide	0.839070002	...	{P1, P3, P6}
⋮	⋮	⋮	⋮		⋮
Cold_15A_0D_R1	AT1G24360.1	Peptide	0.684514579		{P2, P4, P6, P9}
Cold_15A_0D_R1	AT3G26650.1	Peptide	0.76538981	...	{P2, P5, P6}
Cold_15A_0D_R1	AT1G42970.1	Peptide	0.664623548		{P1, P3}
Cold_15A_0D_R1	AT5G48300.1	Peptide	0.644000732		{P2}
⋮	⋮	⋮	⋮		⋮
Cold_180A_0D_R1	AT1G24360.1	Peptide	0.604167075		{P4, P6, P8}
Cold_180A_0D_R1	AT3G26650.1	Peptide	0.70467738	...	{P4}
Cold_180A_0D_R1	AT1G42970.1	Peptide	0.587155655		{P5, P6, P9}
Cold_180A_0D_R1	AT5G48300.1	Peptide	0.54030751		{P1}

Fig. 1. Relational+JSON to capture both structured research data and schema-less contextual information. Highlighted tuples for the *Modification Pattern* $((P_2 \cup P_5) \cap P_6)$.

1.2 Problem Statement and Outline

We developed the problem statement for this paper out of the requirements we gathered from our Computational Biology group who are actively working on integration and exploration of large scale cross-omics[1] research data. Consider a database D with a set of relations $\{R_1, R_2, ..., R_n\}$, each with a JSON column to store an array of attribute-values from known vocabularies. The JSON data adhere to a predefined application schema. Given that the relations follow a hierarchical model $R_1 \rightarrow R_2 \rightarrow ... \rightarrow R_n$ through primary key-foreign key relationships, we want to design a framework that can help data analysts perform exploratory operations on large scale data sets interactively. The framework should allow for progressive filtering of data from the relations of interest while evaluating Boolean predicate expressions on the respective JSON data and let the analyst download a snapshot of the filtered data at any stage of the exploration.

We discuss the related work in the next section and later present our framework with the set of exploration operations that address the given user requirements. We describe in detail our indexing approach for massive sets of attribute-value pairs using Bloom filters, the limitations, and our segmentation approach to overcome those limitations. In our experiments, we compare interactive latencies when using Bloom filter indices with inverted indices and vertical partitioning and conclude by summarizing our experiment results.

2 Related Work and Preliminaries

Decomposed Storage Model (DSM) [2] is one of the early solutions proposed to handle sparse attribute-value format data. Also, extensions of DSM such as Dynamic Tables [3] were used to avoid a large number of relational joins at the time of querying, unlike PIVOT/UNPIVOT operators. The introduction of JSON data types in RDBMS platforms has enabled a document-object-store model by storing JSON data natively without the need for flattening it into relational format [5]. However, the schema-less JSON data can only be considered as text documents for indexing. Typically, a generalized inverted index over JSON data indexes both keywords and hierarchical path structures in a document to provide ad-hoc search capabilities.

Controlled vocabularies play a key role in keeping track of predefined and standardized parameter names. Particularly in bio-science experiments, they add semantics, standard syntax, and formalism to the terms used by scientists to represent them as intended. The finite number of known sets of attributes from the vocabularies combined with the requirement for Boolean containment queries makes Bloom filters the best choice for indexing these huge volumes of contextual data in attribute-value format.

[1] *omics* - an informal reference to a field of study in biology ending in -omics, such as Gen*omics*, Prote*omics*, Metabol*omics* etc.

2.1 Bloom Filters

A Bloom filter [1] is an array of m bits for representing a set $S = \{x_1, x_2, x_3, ..., x_n\}$. To begin with, the bit array is initialized to zero. The idea is to use a set of k uniform hash functions, $h_i(x), 1 \leq i \leq k$ to map elements $x \in S$ to random numbers uniform in the range $1..m$. The basic operations on Bloom filters include adding an element to the set and querying the element membership from the set. To add an element x to the set, it is fed to the k hash-functions and the bits of the returned positions are set to 1. To test if an element is a member of the set, the element is fed to the k hash functions to receive k array positions. The element is not a member of the set if any of the bits at those k positions is 0. This way, the Bloom filter guarantees that there exists *no false negatives*, but it can return *false positives* because of possible hash collisions. Assuming that the hash functions are perfectly random, it is possible to estimate the probability of false positives.

Bloom filters can also be used to perform basic set operations such as union and intersection. Assume that the Bloom filters $BF(S1)$ and $BF(S2)$ represents two separate sets $S1$ and $S2$ respectively.

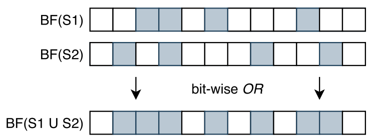


Fig. 2. Bloom filters set union

Definition 1. Set Union*: Considering the same bit array size m and the same number of hash functions k is used in the Bloom filters $BF(S1)$ and $BF(S2)$, then $BF(S1 \cup S2) = BF(S1) \cup BF(S2)$ is represented by a logical OR operation of the two bit arrays. The merged filter $BF(S1 \cup S2)$ will report any element belonging to set $S1$ or $S2$. Also, the false positive probability of $BF(S1) \cup BF(S2)$ is not less than that of $BF(S1)$ and $BF(S2)$ [4].*

Figure 2 depicts the set union operation. The bit-wise logical OR of the respective Bloom filter arrays of the sets $S1$ and $S2$ is sufficient approximation of union that still maintains the original querying property of never returning false negatives.

3 Framework

Within an active user session, each analysis task in our web/client-based framework begins with defining the search space. The framework offers a set of operations with certain semantic constraints to aid data analysts with each step of the interactive analysis. A typical interactive analysis task includes:

1. Defining the search space.
2. Initializing the search space by loading the *signatures* of the data tuples into the main memory.

3. Filtering the data tuple signatures progressively. Allow for interactive predicate construction and evaluation of Boolean predicates.
4. Lazy materialization of the snapshot of the search space at any stage of interactive exploration.

Search Space Definition: The first step of the analysis task is defining the search space which is an imaginary two-dimensional space in main memory. Any relation within the hierarchy can be defined as one of its dimensions or axes. An initial selection predicate can be attached to the dimension definition which gets evaluated at the time of loading the data tuple signatures into the memory. This helps in initializing the search space data structure with the set of data tuples we are interested in. A *dimension* is formally defined as $D_{dim} = \{R, S, P_i\}$ where R is the relation from the hierarchy which is mandatory, S is the signature definition for each data tuple of the relation and P_i is the initial predicate expression. Both signature definition and initial predicate expressions are optional. By default, the *id* of each data tuple of the relation is considered to be its signature.

Initialization: The Search Space data structure is formally defined as $\Omega = \{D_x, D_y, P_x, P_y\}$ where D_x and D_y are the dimension definitions. The framework automatically aligns the dimensions to follow the strict parent-child relation from the hierarchy (not necessarily an immediate parent-child). For example, we cannot have both the dimensions of the search space with the same *Samples* relation for exploration. P_x and P_y are to keep track of sets of attributes in the form of Boolean expressions evaluated on respective attribute-value JSON data. They are initialized to *null* and subsequently updated after each interactive operation. At the time of initialization, the framework joins the two relations of the hierarchy and projects only the *id*s of the respective data tuples which satisfy the given initial predicate expressions in the dimension definitions. Once the search space is initialized, the analyst can execute a range of exploration operations and view the updated search space after every action. The set of operations which can be performed are: *update*, *mockupdate*, *merge*, *save*, *load* and *delete*.

Exploration and Progressive Filtering: The *update* operation takes a dimension name and a set of contextual attributes as its parameters and updates the search space by retaining the signatures of data tuples from the respective dimension containing any of those given attributes. The idea is to use membership queries on the Bloom filter indices and filter out the signatures which do not satisfy the Boolean predicate condition. Consider the initialized search space $S_{sp} = \{D_x, D_y, P_x = \emptyset, P_y = \emptyset\}$. The operation $update(S_{sp}, D_y, [a_1, a_2, ..., a_n])$ would evaluate the Boolean predicate expression $a_1 \cup a_2 \cup ... \cup a_n$ on the dimension D_y and prune the signatures from the search space accordingly, resulting in $S_{sp} = \{D_x, D_y, P_x = (\emptyset), P_y = (a_1 \cup a_2 \cup ... \cup a_n)\}$. Subsequent update operations performed on the search space narrow down the list of signatures to those satisfying the respective Boolean conditions. In summary, the Boolean predicate expressions of subsequent *update* operations on the same search space are equivalent to the logical AND of individual sets of OR-ed attributes. The *merge* operation performs the logical OR on all the signatures from the two search spaces and updates the predicate expressions for each dimension respectively. The *mockupdate* operation works same as *update* operation except that the search space is not updated. It simply returns the summary of the operation result such that the analyst could foresee the result of the *update* operation without actually updating the search space.

Lazy Materialization: One important feature of the framework is the flexibility offered in materializing data tuples and contextual information offline. Upon user command, the full set of data tuples are retrieved from the database using the signatures from the search space and downloaded to disk in the chosen data format, say, CSV, Tab-delimited, JSON, etc. The data analyst can even materialize and download the search space after each exploration operation such as *update* or *merge*.

4 Indexing Using Bloom Filters

Apart from basic set operations, we can also use Bloom filters to query the subsets of elements from the given set of elements. Consider three sets $S = \{x_1, x_2, x_3, ..., x_n\}$, $T = \{x_i : \Phi_1(x), i \in 1..n\}$ and $Q = \{x_j : \Phi_2(x), j \in 1..n\}$ where $T \subseteq S$ and $Q \subseteq S$. The set T is the result set of elements that satisfy a predicate expression $\Phi_1(x)$ and Q is the result set of elements that satisfy a predicate expression $\Phi_2(x)$. Also, the Bloom filters $BF(T)$ and $BF(Q)$ are of same bit array sizes and created using the same number of hash functions. Here, we ignore the false positives for simplicity.

Definition 2. Simple Query*: Checking the membership for all elements of set* S *on* $BF(T)$ *is equivalent to querying the subset of elements from set* S *which satisfy the predicate expression* $\Phi(x)$.

Definition 3. Union Query*: By Definition 1 of* Set Union*, the merged filter* $BF(T \cup Q)$ *reports any element belonging to set* T *or set* Q*. Checking the membership for all elements of set* S *on* $BF(T \cup Q)$ *is equivalent to querying the subset of elements from the set* S *which satisfy the predicate expression* $\Phi_1(x) \cup \Phi_2(x)$.

4.1 Limitations of Bloom Filter Indexing

In addition to false positives, there are a few other limitations of using Bloom filters for set operations. Firstly, when using Bloom filters to query a subset from the given set of elements, we have to test the membership for all elements in the given set. Even though the Bloom filter can answer membership queries in $O(k)$

time for a given bit vector size, the execution time will be proportional to the number of elements to be checked. Secondly, the need for estimating the number of elements in the set beforehand and initialize the bit array. Also, with the growing size of the set, there is a need for periodically testing the false positive rate and the complex task of dynamically re-sizing the filter. Lastly, Bloom filters cannot handle the deletion of elements from sets in general. However, querying subsets of elements from a given set will not have any effect on the result due to deletions. Our indexing approach overcomes these limitations through segmentation and hence not only avoids the need to run through all tuple *id*s in the relation to evaluate the membership query but also can compute the Bloom filter size for a fixed false positive rate.

4.2 Segmentation Approach

Through segmentation, we divide the data tuples in the relation into non-overlapping subsets using a monotonically increasing primary key/tuple identifier. The Bloom filters are created and maintained per segment and attribute as shown in Fig. 3. They are serialized and persisted in secondary storage. Consider a database relation R with a monotonically increasing integer primary key p and a set of attributes $a_1, a_2, ..., a_k$. Let $t_1, t_2, ..., t_n$ be the tuples in the relation R with respective keys $p_1, p_2, ..., p_n$. A segment S_i is a subset of tuples with keys ranging from $p_i, p_{i+1}, ..., p_{i+x}$ in monotonically increasing order where x is the segment size. The first segment S_1 contains the tuples with keys ranging from $p_1, p_2, ..., p_x$. The segments are ordered disjoint sets of keys i.e. $S_1 \cap S_2 \cap ... \cap S_n = \emptyset$ and $S_1 \cup S_2 \cup ... \cup S_n = R$. The segment size x can be optimized. However, it is fixed to a constant size before building the Bloom filter indices since any changes to the segment size requires re-building the indices. We recommend having the segment size multiple folds larger than the total number of attributes to be indexed. This reduces the total number of Bloom filters and retrievals would be faster. However, setting the segment size to a very large number would result in maintaining large-sized Bloom filters. To achieve a fixed bit

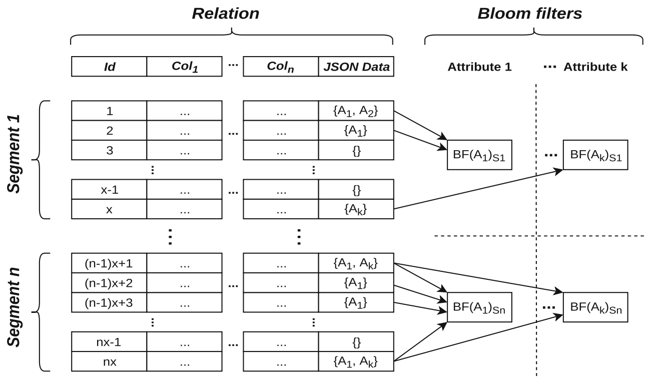


Fig. 3. The segmentation approach

vector size, two other parameters, which can be tuned but have to be set before building the indices are, the number of hash functions and the false positive rate. The segmentation approach is vertically-scalable and as such does not limit the number of attributes to be indexed or the number of data tuples in the relation.

5 Experiments

We compare our Bloom filter indexing with other methods such as Inverted Indices and Vertical Partitioning to evaluate our framework and investigate the following aspects:

1. How fast are interactive response times when Bloom filter indices are used to evaluate Boolean predicate expressions on attribute-value format data?
2. What is the average time taken to perform an end-to-end analysis task?
3. What are the storage space requirements across the three different methods?
4. What is the overall impact of false positives?

We address each of the above points with three databases of varying sizes, say, $Small \approx 50$ GB, $Moderate \approx 104$ GB, and $Large \approx 520$ GB. We use PostgreSQL version 11.7 as the database for all of our experiments performed on a server running Ubuntu 18.04.3 LTS 64 bit. The server hardware configuration is given as 2 $\times$ 6 Core Intel Xeon Bronze 3104 CPU @ 1.70 GHz, 128 GB RAM, and 3TB HDD. The application layer along with the front-end is developed in Spring Boot Framework 2.1.6 and also deployed in the same server.

Data: To simulate a real-world environment, we use uniformly distributed synthetic research data for our evaluation. The test databases for our experiments are created with a simplified hierarchical data model where we only consider two levels of the standardized bio-science research data hierarchy with 8 different types of bio-instances i.e. *lipid, peptide, gene, etc.* We generate a total of 500 Bio-Instance data tuples for each Bio-Sample of any type. Similarly, each Bio-Instance is enriched with 300 randomly chosen attributes from the test vocabulary containing a total of 2550 unique attributes. For example, the *Large* database contains a total of 95 000 Bio-Samples, 47.5 million related Bio-Instances, and a total count of 14.2 billion attribute-value pairs as contextual information. Similarly, the *Small* and *Moderate* databases contain a total of 1.3 and 2.7 billion attribute-value pairs respectively. We use the segmentation approach for both inverted indices and Bloom filters with a fixed segment size of $2^{15} = 32\,768$. The Bloom filters are created with an optimal bit vector size of 314 084 and a total of 7 hash functions to achieve a fixed false positive rate of 1%. For Vertical Partitioning, we create separate relations for each attribute with the *id*s of Bio-Instance tuples containing that attribute as part of its JSON data i.e. a total of 2550 single-column tables, each for one attribute.

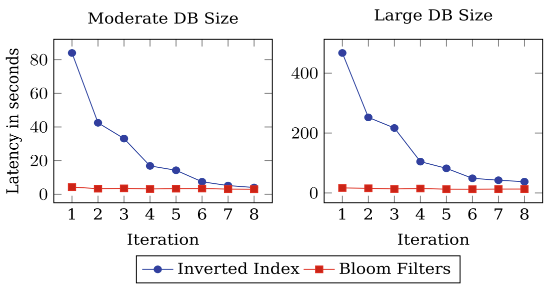


Fig. 4. Latencies measured per iteration

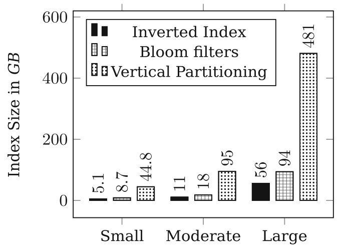


Fig. 5. Index sizes

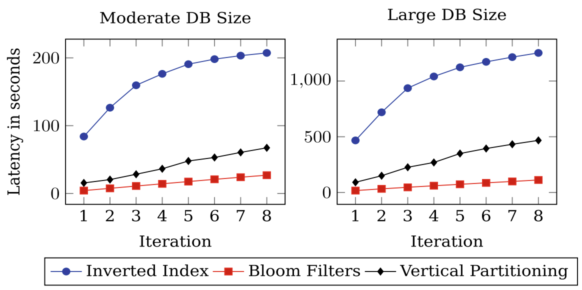


Fig. 6. Cumulative latencies

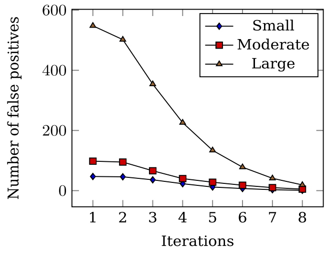


Fig. 7. False positives

Results: We simulated *Modification Patterns* by evaluating our experimental analysis tasks with 8 interactive *update* operations per task evaluated on a set of 5 contextual attributes. The attributes for the experiments are selected randomly out of the test vocabulary and all the analysis tasks are repeated for each bio-instance type (*peptide*, *metabolite* etc.) and the results are averaged for each database. The average number of signatures per each bio-instance type loaded into the memory i.e. the *Search Space* are 0.5 million, 1.1 million and 6 million for databases *Small*, *Moderate* and *Large* respectively. Figure 4 shows the latencies measured at the application layer at the end of each *update* operation with *Moderate* and *Large* databases. We notice a similar pattern with *Small* database as well. When using inverted indices, the early set of *update* operations take more time due to list operations such as sorting, merging, etc. However, as the number of signatures is reduced after each subsequent iteration, the latencies improve. In contrast, irrespective of the size of the search space, Bloom filter indices are much faster with nearly the same latency per each iteration. Figure 6 shows the average cumulative latencies of the interactive operations. We use vertical partitioning as a baseline where we let the database engine perform the union and intersection of *ids*. Here too, using Bloom filter indices proves to be much faster than both the baseline and the inverted indices. It only takes less than 2 min when using Bloom filters to perform all 8 interactive operations on the large search space of 6 million signatures compared to the vertical partitioning and the inverted indices, which take nearly 8 min and more than 20 min respectively. Figure 5 shows the index sizes in GB across different databases measured with the segment size 32 768 to create Bloom filter and inverted indices. Since we used 2550 unique attributes, Vertical Partitioning of *id*s generated the same number of relations. We observe that the inverted indices used less storage space than Bloom filters i.e. ≈10% of the data size vs. ≈17%. However, as expected, Vertical Partitioning required ≈91%. When search space is left with a relatively small number of filtered signatures, the false positives are expected to be near zero. We noticed a similar trend in our experiments and the number of false positives decreased relatively with each *update* operation almost reducing the search space by 50%. Figure 7 shows the decreasing trend of false positives per each iteration with different database sizes.

6 Conclusion and Future Work

In this work, we presented an RDBMS-based vertically-scalable novel framework to interactively explore combined relational and attribute-valued data. We used Bloom filter indices to speed up the interactive response times, particularly, when evaluating Boolean containment queries on a large volume of schema-less data. We demonstrated through our experiments that using Bloom filter indices is many folds faster than the traditional inverted indices while the storage space requirements for the Bloom filter indices are much lower compared to the decomposed storage of the attribute-valued data modeled in JSON format.

As an avenue of future work, we extend the framework for equality and complex predicates such as arbitrary range queries on the attribute-value data. The overall execution times can further be improved by adding a caching layer and implementing the Bloom filter index on JSON datatype within the database layer.

References

1. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970). https://doi.org/10.1145/362686.362692
2. Copeland, G.P., Khoshafian, S.: A decomposition storage model. In: Navathe, S.B. (ed.) Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data, Austin, Texas, USA, 28–31 May 1985. pp. 268–279. ACM Press (1985). https://doi.org/10.1145/318898.318923
3. Corwin, J., Silberschatz, A., Miller, P.L., Marenco, L.N.: Application of information technology: dynamic tables: an architecture for managing evolving, heterogeneous biomedical data in relational database management systems. JAMIA **14**(1), 86–93 (2007). https://doi.org/10.1197/jamia.M2189
4. Guo, D., Wu, J., Chen, H., Yuan, Y., Luo, X.: The dynamic bloom filters. IEEE Trans. Knowl. Data Eng. **22**(1), 120–133 (2010). https://doi.org/10.1109/TKDE.2009.57

5. Liu, Z.H., Hammerschmidt, B.C., McMahon, D.: JSON data management: supporting schema-less development in RDBMS. In: Dyreson, C.E., Li, F., Özsu, M.T. (eds.) International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, 22–27 June 2014. pp. 1247–1258. ACM (2014). https://doi.org/10.1145/2588555.2595628
6. Sansone, S.A., Rocca-Serra, P., Field, D., Maguire, E., Taylor, C., et al.: Toward interoperable bioscience data. Nat. Genet. **44**(2), 121–126 (2012). https://www.nature.com/articles/ng.1054
7. Wang, X., Williams, C., Liu, Z.H., Croghan, J.: Big data management challenges in health research - a literature review. Briefings Bioinform. **20**(1), 156–167 (2019). https://doi.org/10.1093/bib/bbx086

Analyzing Twitter Data with Preferences

Lena Rudenko[1](✉), Christian Haas[1], and Markus Endres[2]

[1] University of Augsburg, Universitätsstr. 6a, 86159 Augsburg, Germany
lena.rudenko@informatik.uni-augsburg.de,
christian.michael.haas@student.uni-augsburg.de
[2] University of Passau, Innstr. 43, 94032 Passau, Germany
markus.endres@uni-passau.de

Abstract. Today Twitter is one of the most important sources for information distribution. But finding useful and interesting tweets on a specific topic is a non-trivial task, because there are thousands of new posts every minute. In this paper, we describe our preference-based search approach on Twitter messages, which allows users to get the best possible results. For this, we introduce a new `CONTAINS` preference constructor to search on full-text data, use NLP techniques to handle natural language mistakes, and present experiments.

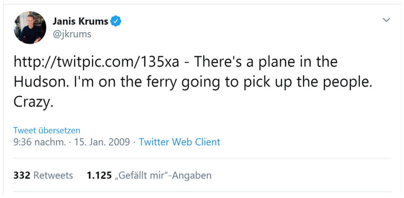
Keywords: Preferences · Twitter · NLP

1 Introduction

Today, social networks and particularly Twitter spread information faster than usual traditional media. The paradigms are changing: first there was the telegraph, then radio and television. Today it is the turn of internet and social networks. They are the fastest way to spread information. For example, the first message about the emergency landing of the flight AWE 1549 in the Hudson River appeared on Twitter less than 5 min after the actual event (see Fig. 1). On Twitter you get the information fast and firsthand. Why watch CNN when you can follow Donald Trump's Twitter? However, it's not Mr. Trump alone, who makes Twitter alive. The emergency landing in Hudson River was reported by the ordinary users with a dozen followers. Every second, about 8.8 thousand messages[1] are posted by Twitter users. Thus, having a source of almost unlimited information, we face the task of finding relevant and interesting content.

On Twitter you can follow the other users and tweets posted by them will show up in your feed automatically. It also allows to browse all tweets using *keywords* or *hashtags*, and quite complex expressions are possible too. This is a great support for users looking for tweets outside the followed accounts, but with one big drawback - if no perfect search matches exist, the user gets no result. An alternative to *hard constraints* are *soft constraints* based on user *preferences*.

[1] Tweets per second: internetlivestats.com/one-second/.



Janis Krums
@jkrums

http://twitpic.com/135xa - There's a plane in the Hudson. I'm on the ferry going to pick up the people. Crazy.

Tweet übersetzen
9:36 nachm. · 15. Jan. 2009 · Twitter Web Client

332 Retweets 1.125 „Gefällt mir"-Angaben

Fig. 1. The first tweet about "Miracle on the Hudson".

Preference-based queries allow the user to specify most preferred and alternative values. However, the challenge in a preference-based text analysis mainly has two aspects: (1) Literally anything can appear in the text. A tweet is limited only by the number of characters. (2) Tweets are very special texts with typing errors, abbreviations, various spelling forms and other special language constructs.

Therefore, it is not surprising that tweet analysis is a popular topic among scientists. One of the research directions, for example, is the analysis of user sentiment and publics' feelings towards certain brand, business, event, etc., cp. [11,15]. A content analysis of tweets on various topics is also often in the focus, cp. [1,2,16]. But none of them deal with Twitter analysis using user preferences and NLP techniques to handle natural language mistakes as we present in this paper.

Our goal is to develop an approach that makes it possible to search for interesting and relevant information among *all* posted tweets using preference-based queries with the guarantee to get no empty result set if perfect hits do not exist. With the expression *text* CONTAINS $(('figure\ skating'), ('isu'))$, for example, the user is looking for all tweets including the term *figure skating*. If no such tweets exist, the messages including *isu* should be returned. But looking for "figure skating", we want to find also tweets with abbreviations or misspelled terms. Some methods from the field of Natural Language Processing could be helpful. To keep the effort of the work within reasonable limits, we only focus on *nouns* in tweets in English for now. The nouns are also expected as input terms for the CONTAINS preference.

The rest of this paper is organized as follows: Sect. 2 recapitulates essential concepts of our *preference model* and *tweet* as a text message. In Sect. 3 we introduce the developed CONTAINS *preference*. Our results on *perliminary experiments* are shown in Sect. 4. Section 5 contains a *summary and outlook*.

2 Background

2.1 Preference Model

Preference modeling have been in focus for some time, leading to diverse approaches, cp. [3,5,6]. These works have in common the treatment of the differentiated user wishes, which prefer some results to the others. We use the approach of [7] that models preferences as strict partial order with intuitiv interpretation *"I like A more than B"*. A preference is defined as $P = (A, <_P)$

on the domain of A. Thus $<_P$ is irreflexive and transitive. Some values are considered to be better than some others and the term $x <_P y$ can be interpreted as *"y is preferred over x"*. If two values cannot be ordered by the strict partial order $<_P$, they are called *indifferent*, i.e. $x \sim_P y \Leftrightarrow \neg(x <_P y) \wedge \neg(y <_p x)$.

The *maximal objects* of a preference $P = (A, <_P)$ on an input dataset R are all tuples that are not dominated by any other tuple w.r.t. the preference. These objects are computed by the preference selection operator $\sigma[P](R)$. It finds all best matching tuples t w.r.t. the preference P: $\sigma[P](R) := \{t \in R | \neg\exists t' \in R : t <_P t'\}$. The evaluation follows a Best-Matches-Only (BMO) query model that retrieves exact matches if such objects exist and best alternatives else.

2.2 Preference Constructors

To express simple preferences targeting one attribute diverse base preference constructors are defined. There are base preference constructors for numerical, categorical, temporal, and spatial domains, cp. [7]. In this work we focus on the categorical preferences. We want to perform a preference-based search on *tweets* that exist as text and thus belong to the categorical attributes.

There exist the following categorical preferences: `LAYERED`, `POS/POS`, `POS/NEG`, `POS`, `NEG`. All categorical base preferences are sub-constructors of $\texttt{LAYERED}_m$ (cp. Fig. 2), which divides the domain into $m + 1$ disjunct sets. Each set consists of i values, where $i >= 1$ and can be different in different sets. The values in one set are considered as *substitutable*. The different sets have different satisfaction levels, whereby the values in the set with $m = 0$ are the *perfect* ones. To be sure that all domain values are covered by the preference constructor, one of the sets may be named *others*.

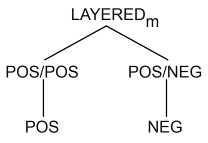


Fig. 2. Taxonomy of categorical base preference constructors

The existing constructors for the categorical attributes work quite well if the domains values are finite, such as book titles, country names, animal species or car colors. A preference query $\text{LAYERED}_{m=1}$`(color, {'white'}, others)` will return all objects that have a value *white* in the attribute *color*. If no such objects exist, objects of any color are returned. This approach will obviously not work with free text attributes, such as *comment*, *note* or *tweet*.

2.3 Tweets

A tweet in general is a complex object with many other attributes besides the actual *text*, such as *language* of the user profile, *number of responses* received by the message and so on. A complete list of all tweet attributes can be found on the Twitter developer website[2]. In this paper, when we use the term *tweet*, we mean a text message only. The numerous attributes that each tweet object includes are not used in this work.

Tweets are short messages with the maximum length of 280 characters, which can contain not only words but also *links*, *special characters* (e.g. emojis), *references* to other user accounts and *hashtags* (keywords or phrases, which help users to find relevant topics). All this is very typical for tweets, which is not surprising when you consider that 83%[3] of all users are those who use the mobile Twitter application and post fast without distracting from other activities.

Not only the content diversity of the tweets, but also various forms of their terms (e.g., *its*, *it is* or *it's* as variants of the same expression) do not allow the existing categorical preferences to provide acceptable results on these text messages. However, based on the existing preference constructors, we decided to develop an additional one that will be able to find the tweets taking user preferences into account.

3 CONTAINS Preference

In this chapter we describe our new `CONTAINS` preference and discuss some techniques from the field of Natural Language Processing we apply to the tweets during the *preprocessing phase* before `CONTAINS` preference can be applied.

3.1 CONTAINS Constructor

Our new `CONTAINS` preference has a similar structure as `LAYERED`. For `CONTAINS` preference a user defines some levels, each with a set of words or terms he would like to see in the result set. The general idea is as follows: if the text message **contains** some term from one of the levels, it gets a number value corresponding to the level number of the term. The smaller the value, the more preferred the tweet is. Finally, the tweets with the smallest values belong to the evaluation result set.

Let $L = (L_1, ..., L_m)$ with $m > 0$ be an ordered list of m sets with terms t_i defined by a user as those he wants to see in the resulted set. Additionally there is also another level with the set called *others*. All terms that will be found in tweets but are not listed in the sets of levels $1, ..., m$ belong to the *others*-set. All sets are disjunct, each term t_i belongs to one set only. The terms within a set are indifferent. Assume x and y are two terms from a set of all terms that occur in tweets. Thus the following applies:

$$x_i <_P y_j,\ x_i \in L_i,\ y_j \in L_j,\ j < i\ for\ i, j \in \{1, ...m+1\} \tag{1}$$

$$x_i \notin L_1, ..., L_m \Rightarrow x_i \in others \tag{2}$$

$$x \wedge y \in L_i \Leftrightarrow (x \sim_P y) \tag{3}$$

[2] Twitter Developer Website: www.developer.twitter.com.

[3] Statista:de.statista.com/statistik/daten/studie/541918/umfrage/anteil-der-mobilen-monatlich-aktive-nutzer-von-twitter-weltweit/.

For each considered tweet a *score value* is calculated. The *lower* the score value for a message, the *more preferred* it is. A tweet can contain terms belonging to *different levels.*

Definition 1. Score Function
Let $t_1, ..., t_n$, $(n > 0)$ *be terms in some tweet* t that belong to some different levels $L_{1,...,m+1}$. The score function is defined as:

$$f(t) = \min(\{i - 1 \mid i = 1, ..., m + 1\})$$

Example 1. The score value for the following tweet $t :=$ *"After @FSUfigure started their campaign against ISU's unfair judging, @olympicchannel removed TES scoreboard from the figure skating live broadcast at the Youth Olympics"* should be calculated. Regarding the preference `P := t CONTAINS` $(('figureskating'), ('isu'))$, t has the score value $f(t) = 0$, even though the text includes terms from levels L_1, L_2 and from *others* (here L_3).

3.2 Natural Language Processing

As mentioned in Sect. 2.3, tweets are short text messages with a high percentage of typing errors, abbreviations and other language features. The new `CONTAINS` preference should be able to find the tweets where the searched terms are written incorrectly or differently. For this we will use some techniques from the field of Natural Language Processing in a preprocessing step.

The following steps are applied to a text before a score value can be calculated with regard to the preference: the tweet text is *normalized*, *segmented* into *sentences*, which are further divided into *single words*. For these words, a *part of speech* is determined and the *nouns* are sorted out of the whole set. The nouns will be stemmed in the last step. A graphic sequence of this process as a pipeline can be seen in Fig. 3.

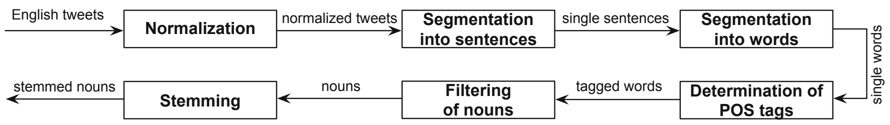


Fig. 3. Pipeline of tweets preprocessing steps.

Let us take a closer look at the individual steps:

1. **Normalization.** In this step, *emojis*, *links*, *retweet-shortcut* (RT at the beginning, if the tweet is a retweet) and the *@username* mentions are removed from the tweet text. In addition, each hashtag term is cleared of the hash character #.

2. **Segmentation into sentences.** After normalization, the text is divided into individual sentences. Although tweets usually contain very few sentences, this segmentation is still important because the POS Tagger determines the word types of the individual words based on the sentence structure.
3. **Segmentation into words.** In this step the sentences are segmented into individual words and passed on to part-of-speech tagger.
4. **Determination of POS tags.** Each input word is assigned to its part of speech with a part-of-speech tagger. Most POS taggers are trained on the data where each word has already been tagged. The trained tagger then assigns a new input word the tag that has the highest probability in the context (cp. [8]). In our implementation we use a Penn Treebank Tagset for English language [9].
5. **Filtering of nouns.** Penn Treebank Tagger distinguishes general nouns and proper nouns in singular and plural. We need all four groups, because, for example, both *"Trump"* and *"president"* can be used in the `CONTAINS` preference. In this step the filtered nouns are additionally cleared of unnecessary characters (e.g. *points* and *apostrophes*), which make stemming problematic.
6. **Stemming.** It will be ensured that the word is in lower case and then a stemming will be performed. Stemming shortens a word back to a word stem common to all morphed forms [14]. Stemming often creates an unnatural common form, but we still use it because it takes no account of the meaning of the word. Since stemming create the unnatural form anyway, it is possible to stem the misspelled words too. The most common stemmer for the English language, which we also use in our work, was published in [13] by M. Porter.

3.3 Edit Distance

The stemmed nouns from the tweets should now be compared with the nouns specified by the user in the `CONTAINS` preference. The latter are also stemmed by the same procedure to enable a fair comparison. Ideally, the word (stem) from a tweet is exactly the same as the word (stem) from a preference. But in order to allow comparison of the misspelled nouns, we calculate the distance between an input stem and a stem from the tweet. The terms with the distance below a certain value are regarded as equal.

In this work we use the **Damerau-Levenshtein** algorithm that works over an *edit distance*. Edit distance between two words is defined as number of changes in one word that have to be made to get from that word to another one. Note that the distance in our work is determined not for the original words, but for their stemmed variants. Changes include *inserting* and *deleting* of a character, *replacing* of a character with another one, and *swapping* the positions of two characters that cover the most common spelling mistakes. Damerau [4] and Peterson [12] independently found out that about 80% of all spelling mistakes belong to one of the four groups: (1) *swap of two adjacent characters* (h**uo**se instead of h**ou**se) (2) *one letter false* (do**k** instead of do**g**) (3) *one letter missing* (a**p**le instead of a**pp**le) and (4) *one letter to many* (informa**i**tion instead of information).

The extended variant of Damerau-Levenshtein algorithm has a *threshold value*. The calculation is aborted if the distance between two words is too big and the threshold value is exceeded. Aborting the distance calculation is possible because we are interested in very similar words. This results in runtime advantages.

To define the term *similar words* we consider three possible scenarios (we used the work of Norvig [10] to decide, which words to consider as **short**, **medium** or **long**):

1. The input word t is **short**: $|t| \leq 4$. The allowed distance d between the short input word and the word from tweets to consider them as equal has to be $d = 0$, so they must be *identical*. Otherwise, there is a risk to identify e.g., *"cat"* and *"cap"* as identical.
2. The input word t has **medium** length: $4 < |t| < 8$. For these words the distance of $d = 1$ is allowed. That means that *one change* is enough to make the input word and tweet word exactly equal. In the words of medium length the probability of making a mistake is much higher than in the short words. At the same time, it is rather improbable to form an existing correct word by making only one change.
3. The input word t is **long**: $|t| \geq 8$. For this group we allow the distance of $d = 2$. It is easy to make a mistake in a long (and often complicated) word. Because of the generally low number of long words, the danger of finding two of them within the distance of $d = 2$ is also low.

To summarize this section: the `CONTAINS` preference allows the user to list the terms that he would like to see in the tweets. A certain deviation in spelling or form, as well as possible errors should be taken into account as much as possible. The texts from the tweets are preprocessed using some methods from Natural Language Processing. Then, both the input terms and the terms from the tweet are stemmed. Finally, the build stems are compared, and if they are equal (which is defined differently for terms of different lengths), the tweet belongs to the result set.

4 Experiments

In this section we describe preliminary test results. These tests should give us the impression of how *efficient* the developed approach is (runtime) and what *quality* the delivered results have (do the results correspond to the query and whether something remains undiscovered).

For comparison we have implemented another method (further called *simpleC*) of score calculation for `CONTAINS` preference. For this, there is no complex natural language tweet preprocessing (but stemming for the preference terms is done anyway). We use a pre-implemented case insensitive function *contains* from the *Apache Commons Lang 3.9* library that checks if one term is contained in another one. For each term in each of preference set it is checked whether it occurs in the tweet and if this is the case, the number of the term's level is assigned to the tweet. The total tweet's score is then the *minimum* of all assigned level numbers (cp. Definition 1).

Thus, *prefC* (our approach) compares stemmed preference terms with the stemmed tweet's nouns. *simpleC*, which we use as a reference point, compares the stemmed preference terms to all original written tweet's words. This decision has to do with the plural form of some English nouns. The word ending *-y* in singular changes into *-ies* in plural, e.g., *study* and *studies*. Thus, when searching for the word *study* in the tweet text, *studies* will not be considered as a hit and vice versa. Spelling mistakes are not taken into account in this method and only the correctly spelled words are regarded as hits.

4.1 Quality Tests

To test the quality of the returned results, 10 000 completely random tweet objects were saved to a text file. As input terms, we used both words that changed their form after stemming and those that remained unchanged. In addition, we selected words of different lengths to be able to test whether wrong spelled words will be found (cp. Sect. 3.3). The two methods are applied to the tweets that we have stored. Table 1 shows the *number* of hits (tweets containing the corresponding term) as well as the same results expressed in terms of ***Precision (P)*** - the percentage of retrieved documents that are relevant to the query - and ***Recall (R)*** - the percentage of the relevant documents that are successfully retrieved.

Table 1. Number of hits, Precision (P) and Recall (R) for two different methods.

Term	Stem	prefC	simpleC	*P* prefC	*P* simpleC	*R* prefC	*R* simpleC
sandwich	sandwich	1	2	100%	100%	50%	100%
housing	hous	35	93	100%	39%	97%	100%
skating	skate	25	1	4%	50%	100%	100%
decision	decis	7	7	100%	100%	100%	100%
connection	connect	8	20	87.5%	40%	78%	89%
replacement	replac	10	15	40%	27%	100%	100%
development	develop	13	19	100%	74%	93%	100%
independence	independ	6	8	50%	37.5%	100%	100%
forest	forest	6	7	100%	100%	85.7%	100%
bear	bear	6	20	100%	30%	100%	100%

The number of hits in Table 1 for the *simpleC* is for most terms higher than that of the *prefC*. There is only one exception for the term *skating*, which we will explain later. *Recall* of *simpleC* is almost everywhere 100%, i.e. *all relevant tweets* have been found. The only exception we observe for the term *connection*.

One of the relevant tweets includes its misspelled form (*konnection*) and will not be found by *simpleC*, but will be present in the results of *prefC*. *prefC* shows slightly worse recall results, but the difference doesn't exceed 15%, except for the term *sandwich*. This term will be foud by *simpleC* in two tweets, while *prefC* returns only one relevant tweet.

The reason for this is that our approach takes into account only a very small number of *hashtags*. They can be almost any - complex, non-existent, invented. Obviously, that stem *sandwich* is found in the hashtag *#tunasandwich* by *simpleC* but not by *prefC*, which can not split this complex term into its components. This is also the case for stem *forest* (cp. Table 1): the seventh hit of *simpleC* is a tweet with the hashtag *#UgandasForests*, which remains undetected by *prefC*.

Besides the "good" results, *simpleC* also detects the "bad" ones in this way: the tweets with hashtags *#clubhousegh* and *#DaquansPlayHouse* are, e.g., returned when the user is looking for *housing*. The tweets delivered by *prefC* (our approach) contain the words: *house*, *housing*, *House*, *house's*, which all make sense. In the tweets delivered with *simpleC*, we find other terms besides those already mentioned, e.g., ***hous**wife*, *pent**hous**e*, *t**hous**and*. Also the proper names (Whitney ***Hous**ton*, Amy *Wine**hous**e*), hashtags (#*uk**hous**ing*, #*club***house***gh*, #*DaquansPlay**Hous**e*) and account names (@*RachaelL**Hous***, @*Alert**Hous**ton* etc.) are included. The result is understandable, because all listed words contain stem *hous*. But it is not what we wanted.

Another reason for the larger result of *simpleC* are *verbs*, *adjectives* and other *nouns* with the corresponding stem (e.g., to ***connect*** and ***connect**ed* while searching for ***connect**ion*; ***bear**y*, to ***bear*** or ***bear**d* - for ***bear***).

The returned non-relevant results have a great effect on the *precision* values, which we can see in Table 1 for the *simpleC* very well. Precision values of *simpleC* are almost always worse, sometimes very clearly than those of *prefC*. The only exception is the term *skating* (see Table 1). The reason is the allowed deviation of the correct spelling/error correction. The original term *skating* is stemmed to *skate*. It has a length of 5, so there is a distance of 1 between *skate* and the stemmed nouns from the tweet allowed (cp. Sect. 3.3). So, e.g., *skate* and *state*, as well as *skate* and *Kate* (the approach is case insensitive) are considered as similar.

Our approach has also found a tweet with a misspelled term *konnection* (instead of *connection*). Unfortunately, the number of incorrectly returned tweets is too high with this correction. So we need to revise this step of our approach. We can increase the length of the words for which correction is allowed. The implementation would be very simple, but we cannot guarantee that even with the longer words, a very similar but completely different term will not appear at the distance of one change.

Another option is to create some kind of lexicon where the most common (spelling) variations of the words are collected. A big advantage of this approach is that we do not have to limit ourself to 4 most common spelling mistakes (cp. Sect. 3.3), but can also include the very popular abbreviations (e.g. *prof* for *professor*). A big disadvantage in return is the effort required to create this lexicon.

In general, we can say that the quality of results delivered with *prefC* is higher compared to *simpleC*. But spelling mistake correction/different spelling part must clearly be rethought and reimplemented.

4.2 Runtime Tests

We also conducted some runtime tests and compared the time for *prefC* vs. *simpleC* to evaluate *one* tweet. The tests were performed on Intel® Xeon® Scalable Processor "Skylake" Silver 4110 with 2.10 GHz, 192 GB DDR4 and 2x 4TB SATA3-HDD. An installed softwaresystem is Ubuntu Linux 18.04 LTS 64 bit.

For the tests we collected real tweets. Each file contains a different number of tweets The tweets were collected live and are different in each file. All tweets in each file were evaluated with respect to two preferences: $\texttt{P}_1\texttt{:=text CONTAINS}(('bear'), ('forest'))$ and $\texttt{P}_2\texttt{:=text CONTAINS}(('housing'), ('decision', 'statement'), ('connection'))$ using *prefC* and *simpleC*. The results can be found in Table 2. The measured time includes both the tweets *preprocessing* and the actual *score calculation*.

Table 2. Runtime of two different methods per tweet in microseconds (μs) for P_1 and P_2.

File	Tweets number	prefC P_1	simpleC P_1	prefC P_2	simpleC P_2
f1	120 277	374.09	108.07	374.40	110.65
f2	133 278	446.79	116.80	449.62	119.62
f3	268 231	438.76	115.39	441.18	118.32
f4	317 816	443.53	116.13	447.05	118.79
f5	399 929	369.35	106.42	370.80	109.17
f6	599 275	365.10	105.30	365.71	107.85
f7	931 027	431.88	113.99	434.53	116.06
f8	1 160 599	435.99	114.41	438.24	117.31
f9	1 307 764	362.31	104.46	362.58	106.48
f10	3 592 899	348.25	104.30	349.26	105.66

As you can see, the runtime of *simpleC* is always significantly better compared to *prefC*. But let us not forget that our approach (*prefA*) requires much more complex preprocessing and score calculation. The other thing that stands out in the results is that regardless of a preference, the runtime for both methods remains very stable and do not depend on the file size (number of tweets).

5 Conclusion

In this paper we described a preference-based approach to analyze tweets. We developed and implemented a new preference constructor named `CONTAINS` that allows users to specify the terms that are preferred in the result set. We have also considered the recognition of misspelled words, because spelling mistakes are very typical for tweets. Our experiments show good *precision* and *recall* values in most cases. We already have some thoughts about improving the runtime while implementing our approach, but the experiments have shown that there is still a lot of work to be done in this direction.

In summary, we believe that the proposed approach has a lot of potential to analyze Twitter data, which we will extend and improve extensively in the future.

References

1. Ayers, J.W., et al.: Why do people use electronic nicotine delivery systems (electronic cigarettes)? A content analysis of Twitter, 2012–2015. PLoS ONE **12**(3), 1–8 (2017)
2. Cavazos-Rehg, P., et al.: A content analysis of depression-related Tweets. Comput. Hum. Behav. **54**, 351–357 (2016)
3. Chomicki, J., Ciaccia, P., Meneghetti, N.: Skyline queries, front and back. SIGMOD **42**(3), 6–18 (2013)
4. Damerau, F.J.: A technique for computer detection and correction of spelling errors. ACM **7**(3), 171–176 (1964)
5. Hristidis, V., Koudas, N., Papakonstantinou, Y.: PREFER: a system for the efficient execution of multi-parametric ranked queries. SIGMOD Rec. **30**(2), 259–270 (2001)
6. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Trade-Offs. Cambridge University Press, Cambridge (1993)
7. Kießling, W.: Foundations of preferences in database systems. In: Proceedings of VLDB 2002, Hong Kong SAR, China, pp. 311–322. VLDB Endowment (2002)
8. Linckels, S., Meinel, C.: Natural language processing. In: E-Librarian Service, pp. 61–79. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-17743-9_4
9. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: the Penn Treebank. Comput. Linguist. **19**(2), 313–330 (1993)
10. Norvig, P.: English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU (2013)
11. Pagolu, V.S., Reddy, K.N., Panda, G., Majhi, B.: Sentiment analysis of Twitter data for predicting stock market movements. In: International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), pp. 1345–1350, October 2016
12. Peterson, J.L.: A note on undetected typing errors. ACM **29**(7), 633–637 (1986)
13. Porter, M.F.: An algorithm for suffix stripping. Program **40**, 211–218 (1980)
14. Samir, A., Lahbib, Z.: Stemming and lemmatization for information retrieval systems in Amazigh language. In: Tabii, Y., Lazaar, M., Al Achhab, M., Enneya, N. (eds.) BDCA 2018. CCIS, vol. 872, pp. 222–233. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96292-4_18

15. Subramaniyaswamy, V., Logesh, R., Abejith, M., Umasankar, S., Umamakeswari, A.: Sentiment analysis of tweets for estimating criticality and security of events. J. Organ. End User Comput. **29**, 51–71 (2017)
16. Sutton, J., et al.: Lung cancer messages on Twitter: content analysis and evaluation. J. Am. Coll. Radiol. **15**, 210–217 (2017)

S-APIR: News-Based Business Sentiment Index

Kazuhiro Seki[1](✉) and Yusuke Ikuta[2]

[1] Konan University, Hyogo, Japan
seki@konan-u.ac.jp
[2] Osaka Sangyo University, Osaka, Japan

Abstract. This paper describes our work on developing a new business sentiment index using daily newspaper articles. We adopt a recurrent neural network (RNN) with Gated Recurrent Units to predict the business sentiment score of a given text and aggregate the scores to define an index, named S-APIR. An RNN is initially trained on Economy Watchers Survey and then fine-tuned on news texts for domain adaptation. Also, a one-class support vector machine is applied to filter out texts irrelevant to business sentiment. Moreover, we propose a simple yet useful approach to temporally analyzing how much and when any given factor influences the predicted business sentiment. The validity and utility of the proposed approach are empirically demonstrated through a series of experiments on Nikkei Newspaper articles published from 2013 to 2018.

Keywords: Deep learning · Sentiment analysis · Text analytics

1 Introduction

There exist business sentiment indices computed through surveys, such as Economy Watchers Survey[1] and Short-term Economic Survey of Principal Enterprise[2] in the case of Japan. These diffusion indices (DI) play a crucial role in decision making for governmental/monetary policies, industrial production planning, and institutional/private investment. However, these DI's rely on traditional surveys, which are costly and time-consuming to conduct.

For example, Economy Watchers Survey is carried out in 12 regions of Japan, where 2,050 preselected respondents who can observe the regional business and economic conditions (e.g., store owners and taxi drivers) fill out a questionnaire and then an investigative organization in each region aggregates the surveys and calculates a DI. As the survey and subsequent processing take time, the DI is published only monthly.

On the other hand, so-called alternative data, including merchandise sales, news, microblogs, query logs, GPS location information, and satellite images,

[1] https://www5.cao.go.jp/keizai3/watcher-e/index-e.html.
[2] https://www.boj.or.jp/en/statistics/tk/long_syu/index.htm/.

J. Darmont et al. (Eds.): ADBIS 2020, CCIS 1259, pp. 189–198, 2020.
https://doi.org/10.1007/978-3-030-54623-6_17

are constantly generated and accumulated. The availability of such data has accelerated the use of data-driven machine learning techniques represented by deep learning for forecasting economic and financial indices [10]. For example, point of sales (POS) data have been used for estimating consumer price index (CPI) [16], financial and economic reports for business sentiment [17], newspaper for grain market prices, stock prices, and economic indices [5,15,18] and social media for stock prices [3,11].

This work focuses on textual data and uses daily newspaper articles to develop a new business sentiment index, named S-APIR index. In addition, using the computed index, we propose a simple approach to temporally analyzing any given factors that may influence the business sentiment index.

2 Related Work

In the economic and financial domains, there are abundant textual data, such as newspaper articles and financial reports in addition to many numerical data. These texts are intended to be read by people, who also consider other sources of information and make decisions on investment, financial policies, and so on. However, it is difficult even for experts to read and grasp all the available information in a limited time. Therefore, there has been much research on computing economical/financial indices from textual data. In the following, we summarize the representative work in business sentiment prediction, which is the main theme of this paper.

Economy Watchers Survey introduced in Sect. 1 publishes not only the business sentiment index (hereafter called EWDI for short) but also individual survey responses on which EWDI is based. The survey responses contain a pair of an economic condition on a five-point scale and a statement of the reasons in natural language why the respondent chose a particular economic condition.

Based on the responses, EWDI is computed by first computing the composition ratios of the five economic conditions (×, △, □, ○, ◎) and then taking their weighted sum. EWDI ranges from 0 to 100 with 50 being the middle (50 means that economic condition is neither positive or negative).

Yamamoto et al. [17] used as training data around 200,000 pairs of an economic condition and its statement of the reasons to learn a regression model so as to predict the business sentiment of a given text. As a regression model, they used a bidirectional Recurrent Neural Network with Long Short Term Memory (LSTM) [9]. Then, monthly economic reports were fed to the learned model to compute a business sentiment index. It is reported that the computed index was positively correlated with both EWDI and Short-term Economic Survey of Principal Enterprise in Japan. Similarly, Aiba et al. [1] used the same model to compute a business sentiment index from microblogs (tweets). Goshima et al. [8] adopted a convolutional neural network and used Reuters news articles as input to compute a business sentiment index.

3 S-APIR and Its Application to Word-Level Analysis

As described in Sect. 2, Economy Watchers Survey contains pairs of an economic condition and a statement of the reasons. They are filled out manually by respondents and thus are quality and valuable resources for machine learning. In the present work, we focus on news articles as with Goshima et al. [8] to compute a business sentiment index named S-APIR. However, in contrast to Goshima et al. who fed all the available news texts to the learned model, we attempt to filter out irrelevant news texts and then to apply domain adaptation as described shortly in Sect. 3.1.

In addition, Sect. 3.2 discusses an application of the S-APIR index to temporally analyze any given factors that may/may not influence business sentiment. Business sentiment is formed by many factors including monetary policies, stock prices, exchange rates, unemployment rate, wages, overseas situations, etc. However, those factors do not equally influence business sentiment and it is helpful for business economists if they could understand what factors have a more/less effect to improve or decline business sentiment in a particular period. To this end, we propose a simple approach to analyzing when, how much, and what factors contributed to S-APIR based on the predicted business sentiment.

3.1 S-APIR Index

This section describes three major components of our framework to predict business sentiment from a news sentence. The first component is a regression model that takes a sentence and predicts the sentiment of the input. The second is a classifier to filter out texts irrelevant to the economy/business. Lastly, the third is domain adaptation to update the parameters of an initial regression model to make them more suitable for news texts.

Regression Model. For text classification and regression, it has been commonly done to treat each word as an independent variable, where an input text is represented as a Bag of Words (BoW) disregarding the context [13]. However, it is desirable to capture the differences of word meanings in different contexts and word dependencies so as to properly represent the meaning of the text.

In recent years, RNN, combined with LSTM, has been popularly used to represent text to consider the context. This study also uses RNN but with Gated Recurrent Unit (GRU) [6], which can be seen as a variant of LSTM. Following the related work, we also use Economy Watchers Survey to train the model, where the five-point-scale economic conditions $\{\times, \triangle, \square, \bigcirc, \circledcirc\}$ are converted to $\{-2, -1, 0, 1, 2\}$, respectively.

Note that other models including bidirectional LSTM-RNN, bidirectional GRU-RNN, and Bidirectional Encoder Representations from Transformers (BERT) [7] with fine-tuning could be used for prediction. In fact, we tested these models but the simple, one-directional GRU-RNN generally yielded better results for this task and we will omit the results of the other models in Sect. 4.

Filtering. The regression model described in the previous section can be used to predict business sentiment for any input text. However, news texts we focus on in this study are in many genres (e.g., sports) which may be irrelevant to the economy. Using irrelevant sources would be harmful in computing a business sentiment index. Therefore, we attempt to filter out such irrelevant news texts by treating them as outliers.

For this purpose, we adopt a one-class support vector machine (SVM) [12]. In contrast to an ordinal SVM used for binary classification, a one-class SVM can be learned on documents in only one class and detect documents dissimilar to the training documents as outliers. We use Economy Watchers Survey (specifically, statements of the reasons) as the training data for one-class SVM and filter out news text dissimilar to the statements.

For text representation, the one-class SVM uses the traditional BoW with term frequency-inverted document frequency (tf-idf) term weighting [13]. One could use an output of an RNN or other sentence embeddings [4,14] so that the context could be better represented. However, our preliminary experiment showed that they resulted in detecting all news texts as outliers and were not used in the present work.

Domain Adaptation. The Economy Watchers Survey responses to be used for training a GRU-RNN are different from news texts to be used for computing S-APIR in terms of their writing styles, vocabularies, and expressions (here collectively called "domains"). Such differences between training and testing would have a negative effect on the resulting performance, and adapting the domain of the learned model to the target domain would benefit business sentiment prediction.

To this end, we explore domain adaptation by automatically creating new training data from news texts. To be precise, we feed news articles to an initial regression model (denoted as M) and predict the business sentiment of each sentence. Then, we assume that sentences with higher absolute sentiment scores would better represent economic conditions either positively or negatively. We set predefined positive and negative thresholds and treat the sentences with higher/lower sentiment scores than the thresholds as positive and negative examples, respectively.

We use the training data to fine-tune the initial model M to acquire a fine-tuned model M'. More specific experimental settings (e.g., thresholds) are described in Sect. 4.

3.2 Word-Level Temporal Analysis

Business sentiment is formed by various factors including monetary policies, trade, military conflicts, an outbreak of pandemic diseases, and so on. We propose a simple approach to analyzing *which* factor influenced business sentiment *when* and to *what degree*. Specifically, we define the influence of word w during time period t, denoted as $p_{t,w}$, using the predicted business sentiment.

We first assume that the sentiment p_s of a sentence s is the sum of the sentiments of words (w) appearing in s as follows:

$$p_s = \sum_{w \in s} f_{s,w} \cdot p_{s,w} \tag{1}$$

where $f_{s,w}$ is the number of occurrences of word w in s, $p_{s,w}$ is the sentiment of w in s. We further assume that all the words, $w \in s$, equally influence the sentiment of s, that is, $p_{s,w} = p_s/|s|$, where $|s|$ is the number of words composing s. Here, let S_t denote the set of news sentences published during t. Using S_t, we define $p_{t,w}$ as the sum of $p_{s,w}$ over S_t, divided by the number of sentences $|S_t|$, that is,

$$p_{t,w} = \frac{1}{|S_t|} \sum_{s \in S_t} f_{s,w} \cdot \frac{p_s}{|s|} \ . \tag{2}$$

Intuitively, S-APIR during t can be interpreted as the sum of the influences of all the words appearing in texts published during t.

4 Evaluation

4.1 Experimental Settings

For learning an RNN and a one-class SVM, we downloaded the Economy Watchers Survey data from the web page of the Cabinet Office[3] in October 2018. The number of the pairs of an economic condition and a statement of the reasons was 216,741 in total, of which randomly selected 90% were used for training (and validation) and the rest were used for testing. Note that because Japanese text does not have explicit word boundaries (such as spaces for English), the statements of the reasons were processed by a morphological analyzer, MeCab[4], to be split into words.

The parameters of a GRU-RNN were set as follows based on a preliminary experiment on the Economy Watchers Survey data: the number of GRU units per layer = 512, the number of hidden layers = 2, and the size of the vocabulary = 40,000. Each word was represented as a word embedding vector with 300 dimensions pretrained on Wikipedia [2].

To compute the S-APIR index, we used the titles and body texts of news articles from the Nikkei Newspaper from 2013 to 2018. For domain adaptation, we used separate Nikkei Newspaper published in 2010. Each article was split into sentences based on the Japanese period "。" and each sentence was fed to the learned model to predict its sentiment.

4.2 Evaluation on Economy Watchers Survey

First, we evaluated the initial GRU-RNN on the held-out test data (i.e., 10% of Economy Watchers Survey) in mean squared error (MSE). For comparison, we

3 http://www5.cao.go.jp/keizai3/watcher/watcher_menu.html.
4 http://taku910.github.io/mecab/.

also applied a ridge regression model with the classic BoW representation with tf-idf term weighting. MSE was found to be 0.351 for GRU-RNN and 0.509 for ridge regression, which confirms that our model, GRU-RNN, predicted economic conditions more accurately than the ridge regression. This result is similar to the one reported in the related work [17].

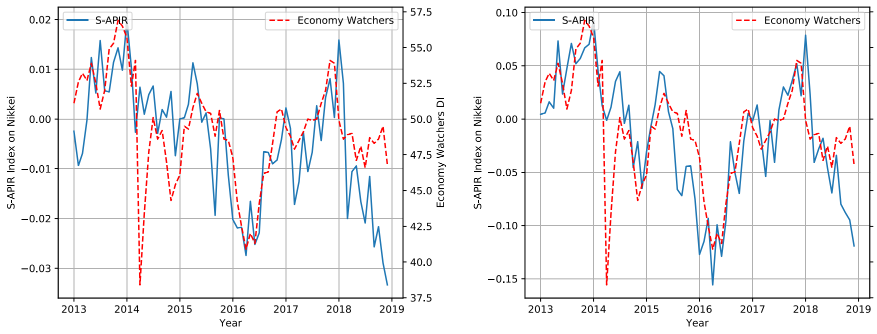


Fig. 1. Comparison between S-APIR and EWDI ($r = 0.546$) (left) and between S-APIR and EWDI after domain adaptation ($r = 0.701$) (right).

4.3 Evaluation of S-APIR

This section compares our business sentiment index, S-APIR, and existing business sentiment index, namely EWDI. However, it should be emphasized that S-APIR is not intended to replace EWDI, but rather to be a new index using newspaper as the source of information. There is no ground truth for a business sentiment index and EWDI is also one of possible indices, which is calculated based on the limited, 2,050 respondents. Nevertheless, we make the comparison (a) to ensure that S-APIR generally has a similar trend to the established, existing index and (b) to investigate the characteristics of S-APIR when the two indices diverge.

Results. Using the initial GRU-RNN learned as described in Sect. 4.2, we first computed S-APIR on Nikkei Newspaper from 2013 to 2018. Note that as business sentiment is predicted for each sentence, they were aggregated monthly by taking the average so that S-APIR could be directly compared with EWDI. Figure 1 (left) shows the computed S-APIR and EWDI for comparison. We can observe that the two indices show roughly similar movements. In effect, they were found to be positively correlated ($r = 0.546$).

Next, we applied the one-class SVM to detect and filter out outliers and recomputed S-APIR by using only the texts relevant to the economy. Overall, S-APIR exhibited a more similar trend to EWDI and their correlation coefficient significantly increased from 0.546 to 0.686 (their plots are not shown due to the page limit). The result confirms the effectiveness of the filtering process by the one-class SVM.

As shown above, the GRU-RNN learned on Economy Watchers Survey (statements of the reasons) can be used to compute the business sentiment index based on news texts. However, statements of the reasons and news texts have different characteristics and the model learned on the former may not be suitable for the latter. Thus, we applied domain adaptation as described in Sect. 3.1. To be precise, we took the following procedure:

1. Extracted titles and body texts of news articles from Nikkei Newspaper 2010 and split them into sentences and then into words.
2. Applied the one-class SVM and filtered out outliers.
3. Applied the GRU-RNN to predict the sentiment of each sentence.
4. Identified the sentences with sentiment scores greater (lower) than a predefined threshold t_{high} (t_{low}). To determine the thresholds, we looked at the histogram of the sentiment scores and experimentally set $t_{high} = 0.8$ and $t_{low} = -1.0$. As a result, we obtained 18,947 positive instances and 10,868 negative instances. We gave the former "2" as their labels, and the latter "−2".
5. Used the generated training data to fine-tune the initial GRU-RNN.

Then, we used the fine-tuned model to recompute S-APIR after filtering. The result is shown in Fig. 1 (right). The correlation coefficient further but marginally increased to 0.701.

Inspecting the resulting plots in Fig. 1, we observed that EWDI dropped sharply in April 2014, where there is a large deviation from the S-APIR index. This is when sales tax was increased from 5% to 8% in Japan and it is interesting to find that the S-APIR index is much less affected by the tax increase. We conjectured that this may be due to the fact that 70% of the respondents of Economy Watchers Survey had occupations related to households. Therefore, factors that have more influence on households (e.g., tax increase) may have more influence on EWDI as well. To verify the intuition, we compared S-APIR and a variant of EWDI computed based on responses only from those who have occupations related to industries. As a result, the correlation coefficient significantly increased from 0.701 to 0.819. This result implies a property of the S-APIR index computed from Nikkei Newspaper that it reflects business sentiment in industries more strongly.

Effect of Domain Adaptation. The previous section empirically showed that domain adaptation increased the correlation between S-APIR and EWDI but only marginally. Here, we look into the models before/after domain adaptation to investigate how the model changed. Specifically, we fed single words as inputs to each model and predict the sentiments of the individual words.

Table 1 compares ten words with higher sentiment for each model on descending order of the sentiment scores, where words with "↑" indicate those went up after domain adaptation and those with "↓" indicate went down. While the rankings of "好調 (good condition)", "享受 (to enjoy)", "回復 (recovery)", and others went up, those of "最高 (best)", "絶好調 (best condition)", and "伸 (to grow)" went down.

Table 1. Business sentiment of top 10 positive words before/after domain adaptation.

Before		After		
快調 (good condition)	0.738	↑	好調 (good condition)	1.591
絶好調 (best condition)	0.688	↑	享受 (to enjoy)	1.503
好調 (good condition)	0.671	↑	復調 (recovery)	1.502
最高 (best)	0.659	↑	好転 (to improve)	1.488
上々 (excellent)	0.654	↑	向上 (to improve)	1.485
好転 (to improve)	0.654	↓	最高 (best)	1.484
伸 (to grow)	0.632	↑	刺激 (stimulation)	1.469
売れれ (to sell)	0.630	↓	絶好調 (best condition)	1.456
上向い (to look up)	0.624	↓	伸 (to grow)	1.449
着実 (steady)	0.617	↑	図り (to plan)	1.437

Similarly, we compared ten words with lower sentiment scores (the resulting table is not shown due to the page limit). In both cases, we observed that words that would be more often used in news text went up and those used more often in survey responses went down after domain adaptation.

Notice that while we predicted business sentiments of individual words to investigate how the model changed after domain adaptation, resulting business sentiment scores can be seen as business sentiment polarities of the words. That is, the results can be used as a sentiment dictionary in the economic/financial domain.

4.4 Temporal Analysis of Words

Lastly, we temporally analyzed the influence of a given factor (word) on S-APIR as described in Sect. 3.2. While our proposed approach can analyze any given words, here we focused on a few representative examples. Specifically, we computed the influence of "中国 (China)" and "貿易 (trade)" for example. The results are shown in Fig. 2, where the upper plots show S-APIR from Fig. 1 (right) for reference.

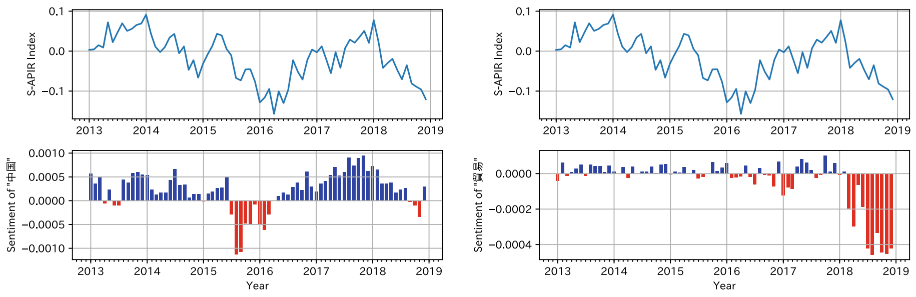


Fig. 2. Temporal influence of "中国 (China)" on S-APIR (left) and that of "貿易 (trade)" on S-APIR (right).

In Fig. 2 (left), S-APIR and the influence of China generally have similar movements and thus situations about China appear to be one of the major factors influencing the business sentiment index in Japan. Especially, from the middle of 2015 to the beginning of 2016, the influence of China is strongly negative, which is pushing down the business sentiment in Japan. This period is a time when the Chinese economy deteriorated rapidly due to the crash of China's stock market.

Then, looking into Fig. 2 (right), we can observe that "trade" did not have much influence from 2013 to the beginning of 2018, whereas the situation has changed thereafter and started to show a strong negative influence on business sentiment. This reflects the US-China trade dispute that began in late 2018.

5 Conclusions

This paper reported on our ongoing work to develop a new business sentiment index, called S-APIR, based on news texts and to use the index to temporally analyze the factors that influence business sentiment. We used a one-class SVM to identify news texts related to the economy and fed them to a GRU-RNN regression model to predict the business sentiment of input news text. The GRU-RNN was initially trained on Economy Watchers Survey and then fine-tuned on news texts for domain adaptation. Through our evaluation using Nikkei Newspaper articles, it was demonstrated that S-APIR has a positive correlation with an existing business sentiment index and that the correlation becomes even higher when compared to the index calculated only from responses in industries. This result indicates that S-APIR is a business sentiment index reflecting that of industries more strongly. Moreover, by dividing sentence sentiment into word sentiments and summing over sentences, it was shown that any given factor that may/may not have an influence on business sentiment can be temporally analyzed and visualized.

Acknowledgments. This work was conducted partly as a research project "development and application of new business sentiment index based on textual data" at APIR and was partially supported by JSPS KAKENHI #JP18K11558 and MEXT, Japan. We thank Hideo Miyahara, Hiroshi Iwano, Yuzo Honda, Yoshihisa Inada, Yoichi Matsubayashi, and Akira Nakayama for their support. The Nikkei data were provided by APIR.

References

1. Aiba, Y., Yamamoto, H.: Data science and new financial engineering. Bus. Observ. **81**(2), 30–41 (2018). (in Japanese)
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
3. Bollen, J., Mao, H., Zeng, X.: Twitter mood predicts the stock market. J. Comput. Sci. **2**(1), 1–8 (2011)
4. Cer, D., et al.: Universal sentence encoder for English. In: Proceedings of the 2018 EMNLP, pp. 169–174 (2018)

5. Chakraborty, S., Venkataraman, A., Jagabathula, S., Subramanian, L.: Predicting socio-economic indicators using news events. In: Proceedings of the 22nd ACM SIGKDD, pp. 1455–1464 (2016)
6. Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. In: Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation, pp. 103–111 (2014)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 NAACL-HLT, pp. 4171–4186 (2019)
8. Goshima, K., Takahashi, D., Yamada, T.: Construction of business news index by natural language processing and its application to volatility prediction. Finan. Res. **38**(3), 1–41 (2019). (In Japanese)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Kapetanios, G., Papailias, F.: Big data & macroeconomic nowcasting: Methodological review. Economic Statistics Centre of Excellence (ESCoE) Discussion Papers ESCoE DP-2018-12, Economic Statistics Centre of Excellence (ESCoE) (2018)
11. Levenberg, A., Pulman, S., Moilanen, K., Simpson, E., Roberts, S.: Predicting economic indicators from web text using sentiment composition. Int. J. Comput. Commun. Eng. **3**(2), 109–115 (2014)
12. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. J. Mach. Learn. Res. **2**, 139–154 (2002)
13. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
14. Pagliardini, M., Gupta, P., Jaggi, M.: Unsupervised learning of sentence embeddings using compositional n-gram features. In: Proceedings of the 16th NAACL, pp. 528–540 (2018)
15. Shapiro, A.H., Sudhof, M., Wilson, D.: Measuring news sentiment. Federal Reserve Bank of San Francisco Working Paper 2017–01, Federal Reserve Bank of San Francisco (2017)
16. Watanabe, K., Watanabe, T.: Estimating daily inflation using scanner data: A progress report. In: CARF Working Paper Series. No. CARF-F-342 (2014)
17. Yamamoto, Y., Matsuo, Y.: Sentiment summarization of financial reports by LSTM RNN model with the Japan economic watcher survey data. In: Proceedings of the 30th JSAI (2016). (In Japanese)
18. Yoshihara, A., Seki, K., Uehara, K.: Leveraging temporal properties of news events for stock market prediction. Artif. Intell. Res. **5**(1), 103–110 (2016)

Towards an Inference Detection System Against Multi-database Attacks

Paul Lachat[2,3](✉), Veronika Rehn-Sonigo[1](✉), and Nadia Bennani[2](✉)

[1] University of Bourgogne Franche-Comte, FEMTO-ST Institute, CNRS, Besançon, France
veronika.sonigo@femto-st.fr

[2] University of Lyon, CNRS, INSA Lyon, LIRIS, Lyon, France
{paul.lachat,nadia.bennani}@insa-lyon.fr

[3] Department of Distributed and Multimedia Information Systems, University of Passau, Passau, Germany

Abstract. Nowadays, users are permanently prompted to create web accounts when they buy online goods. This collected data gives an insight on the user, sometimes beyond the application scope. Inference attacks on databases represent an issue for data controllers when malicious processors attempt to guess sensitive data - to which they haven't access - by inferring them using legally accessed data. Several inference attack detection systems address this problem in case of a single targeted database. But the issue remains unsolved in case of several databases to which the same users might have submitted their data. In this paper, we propose a global model and its associated graph representation named *Global Instance Graph* (GIG) representing the probabilistic and semantic dependencies inside each database, enriched by the dependencies between the different databases. The graph is obtained using privacy-preserving record linkage techniques and serves as a knowledge input to the inference attack detection system. We validate the GIG creation feasibility thanks to a proof of concept. Despite the quadratic creation time, the performances when data is queried from the databases are not affected since the GIG creation is performed offline.

Keywords: Inference detection · Multi-database attacks · Data privacy · Privacy-preserving record linkage

1 Introduction

We are assisting an era with almost unlimited data storage capacities thanks to the constant increase in data centers' offers. As a consequence, online applications do not hesitate to store huge amounts of raw data concerning their users' habits and behaviors. Moreover, users are permanently prompted to disclose more information including their personal data. But sometimes, this collected data gives an insight on the user beyond the application scope. This could benefit to external stakeholders with the objective to learn more about the user, which could harm the user's privacy. New regulations are now emerging to protect the user privacy, among them the *General Data Protection Regulation* (GDPR) which is intended to regulate the way EU citizens' personal data should be consumed (according to user's consent) by data controllers and data processors. Moreover the GDPR is not only a theoretic regulation: Ongoing research works on concrete implementations, e.g. [6]. In fact, user's privacy can be disclosed via inference attacks where malicious processors and controllers exploit the possibility to indirectly infer sensitive data to which they haven't access thanks to legally accessed data. This problem can be locally solved using inference systems proposed in the literature [1–3,7,10]. However these solutions are not suitable in case of inference attempts where several databases are implied.

Let us consider the following scenario in which a user u registers on two online applications: $Train$ to buy train tickets and $Flight$ for flight tickets. For each application, u gives a set of her personal data to validate the registration process. At this stage, $Train$ and $Flight$ have the ability to protect the user data thanks to: the implemented access control policies and a locally implemented inference attack detection system. Moreover each application is allowed to share with other controllers and processors, with the consent of u, a part of her collected personal data. In this scenario, we could imagine a common processor $Service$ which gather and process travel information coming from several tour operators (e.g., $Flight$ and $Train$). In this case, $Service$ has legal access to subsets of personal data coming from both operators (among them u's personal data subset F_u and T_u from $Flight$ and $Train$ respectively). This creates possible inference opportunities: a part of F_u could be exploited by $Service$ in order to reveal u's sensitive personal data in T_u while bypassing the local inference detection system of $Train$. This scenario highlights the exploitation of inference channels targeting multiple databases through legal access by the same entity. Inference attacks exploiting these inference channels rely on the *distributed dependency strategy* [12]. We will refer to them in the remainder of the paper as *Multiple Database Inference Attack* (MDIA).

Inference attacks happen either in a non-interactive setting (e.g. dump from a database, log files, etc.) or in an interactive setting (e.g. by means of Web browser, localization systems, database queries, and so on). In this paper we focus on the later and thus we give a brief description of the related proposals in the scientific literature. Staddon [10] presents a solution where keys are given to the users in order to generate the required tokens to query the objects in an inference channel and thus prevent its full exploitation. The solution presented by Biskup [1] relies on dynamically adapting policies preventing malicious users to fully exploit inference channels in a logic-oriented information system. The system presented by Brodsky et al. [2] models the functional dependencies of a database to compute the disclosed knowledge each time a query is issued. Then, based on the query log of the issuing user, the system either denies the answer or returns a fake one. Guarnieri et al. [7] propose a system where one module acts as a policy decision point whereas the other checks inference attempts. The latter relies on the security policy defining the inference threshold of the sensitive information and the attacker model describing the user's *a priori* knowledge. Then, based on the inference detection results, the first module decides to deliver or not the query answer. Chen et al. [3] propose to tackle inference attacks by building first a *Semantic Inference Model* (SIM) representing the probability of attributes to influence others. Then based on the SIM dependencies, a *Semantic Instance Graph* (SIG) reflecting these dependencies at the instances' level in the database is generated and enrolled to detect the inference attacks. All these works present solutions capable of preventing inference attacks on single databases, but they are not adapted for inference attacks when multiple databases are involved.

In this paper we propose an extension of the work of Chen et al. [3] to address the MDIA issue. Our proposal consists in building a *Global Instance Graph* (GIG) by discovering similarities between instances in the SIGs from the concerned databases in order to model inference channels that could be exploited by MDIAs. Moreover, to avoid honest-but-curious behavior, the SIGs are first anonymised. The faced challenge is to discover these similarities on anonymised data. This paper presents the following contributions: (**i**) The GIG creation algorithm based on a set of SIGs, using Bloom Filters [8] to discover instance similarities while preserving the users' data privacy. **(ii)** An architecture that manages the MDIAs for a set of implied databases.

2 Inference Detection in Case of a Single Database

In this section we briefly review the approach of Chen which is based on the creation of a *Semantic Inference Model* (SIM). For more details please refer to the original paper. The purpose of the SIM is to represent the inference channels of a database at schema level. It extends the *Probabilistic Relational Model* (PRM) which is based on Bayesian networks. According to [5], the PRM "[...] allows the properties of an object to depend probabilistically both on other properties of that object and on properties of *related* objects". It is made up of two parts: a skeleton and the parameter. The skeleton represents the relations between each attribute and his *parents* which have a direct influence on it. The parameter is the *Conditional Probability Distribution* (CPD) between an attribute and his *parents*. The CPD is computed for a given database and thus represents the distribution of the data at the time of computation. The SIM is composed of three types of links: (i) dependency links: which are the dependencies related to the skeleton of the PRM (ii) schema links: which connect primary keys and foreign keys in the databases tables and (iii) semantic links: which represent dependencies that can be provided by an operator with domain specific knowledge or by analyzing queries issued to the database. The SIM is instantiated with the instances of the database into a *Semantic Instance Graph* (SIG) in order to represent the dependencies at the instance-level. Thus, the nodes in a SIG represent the attribute values of a specific instance in a database. To protect data against inference attacks, sensitive attributes are identified and their inference thresholds assigned. An inference attack is detected once the percentage of confidence about the value of a sensitive attribute exceeds his corresponding threshold. A Bayesian network is instantiated from the SIG for each user in order to keep track of the knowledge she has about the database.

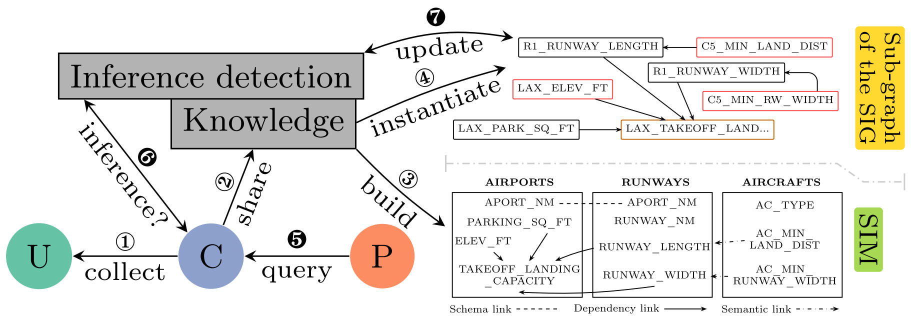


Fig. 1. Workflow of the Chen et al. solution.

The following example, extracted from [3], illustrates the overall workflow of their architecture and the inference attacks detection process. The white steps (i.e., ① to ④) in Fig. 1 are made once offline. They represent the SIM and the SIG computing. Whereas the black steps (i.e., ❺ to ❼) are executed whenever a query is submitted to the system. The example represents a chunk of a SIG related to three instances in the database: *LAX*, *R1* and *C5*. For clarity, each attribute instance in the SIG is prefixed with the name of the instance to which it belongs. Thus, multiple occurrences of the attribute RUNWAY_WIDTH could be related to different instances. The example assumes that the attribute TAKEOFF_LANDING_CAPACITY (TLC) is defined as a sensitive attribute and his inference threshold is set to 70%. If a user knows that the instance *C5* is able to land on *R1* from *LAX* and that the instance attributes C5_MIN_LAND_DIST and C5_MIN_RUNWAY_WIDTH have `long` and `wide` values respectively. Then the user can infer the value for TLC with a confidence of 58.3%. If the same user succeeds to obtain the value of the instance attribute LAX_PARKING_SQ_FT (which is `large`), then she can infer the value `'large'` for the instance attribute TLC with a confidence of 71.5% which is above the inference threshold of TLC. Therefore, answering the last query will lead to an inference attack. This solution works well when the queries target the same protected database. However it does not prevent malicious users to query the data required to infer sensitive values both from the protected database and from an external source. This latter scenario is not detected by such a system.

3 Extension to a Multi-database Protection: A Step Ahead

Multi Database Inference Attacks (MDIA) occur due to the fact that personal information is scattered among several databases and sensitive data can then be inferred using the knowledge obtained from each database separately. Thus, traditional inference detection systems like [1–3,7,10] are not able to catch such attacks since they only model the inference channels present in a single database. To tackle this issue, our approach proposes to aggregate data controllers' SIGs into one global model called the *Global Instance Graph* (GIG) to be able to represent both inference channels within each database and those implied by the access to several databases. Solving the MDIA issue is not an easy task, we make the following set of hypothesis that remain realistic: (i) We assume that each data controller interested in protecting its users' privacy subscribes to a provider that proposes such a solution and collaborates with it. (ii) The data controllers do not send in clear user data to the inference detection module. (iii) The proposed inference detection module is centralized in order to reuse the inference detection algorithm of Chen by replacing the input SIGs with the GIG. (iv) The inference detection module does not collude with any of the data controllers that subscribe to use its service, but we assume that is could have an honest-but-curious behavior. (v) The databases managed by data controllers are not subject to updates. (vi) The data processors do not collude.

To compute the GIG, one must identify similar instances present in different SIGs. Instances are said to be similar if they represent the same real world entity. For example the instances: (*Alice*, *Thing*, *1992-07-12*) and (*Alice Thing*, *12/7/1992*) have different formats but represent the same real world entity. As a consequence, the GIG must represent those relations of similarity by adding a new kind of links, the *similarity links*, between nodes of similar instances in different SIGs. Adding such links allows to model the propagation of a user knowledge beyond the SIG of the queried database to the other SIGs. In other words, if in the GIG, the nodes n_1 and n_2 from different SIGs are both linked with a *similarity link*, then if a user queries the value of n_1 her knowledge of this attribute value is set to 100% thus the probabilistic propagation will set the percentage of knowledge of the n_2 value to 100%. Therefore, the main challenge of computing the *similarity links* is related to the data format heterogeneity among databases. As demonstrated in [8] *Bloom Filter* (BF) is the most commonly used structure when calculating similarity scores (e.g., based on the *Dice coefficient*). The second challenge to respect hypothesis (ii) is to anonymize data in the SIG before sending it to the inference detection module. To keep the similarity calculation possible, the anonymization function used on each SIG must be the same. The BF must also preserve the privacy of the encoded instances. Encoding techniques such as the one demonstrated in [8] are sensible to attacks, based on frequency accounting or bit pattern, aiming to re-identify data encoded within the BF structure. Such an attack is presented in [4] where the following recommendations to use BF for preserving-privacy computing is proposed: (i) use record-level[1] BF encoding (like the CLK approach proposed in [9]), (ii) employ different hash mechanisms, and, (iii) use advanced techniques (random hashing, adding random bits, etc.).

[1] Record-level mean that each field (i.e., attribute) of an instance are encoded into a single BF whereas field-level mean that each field is encoded into a separated BF.

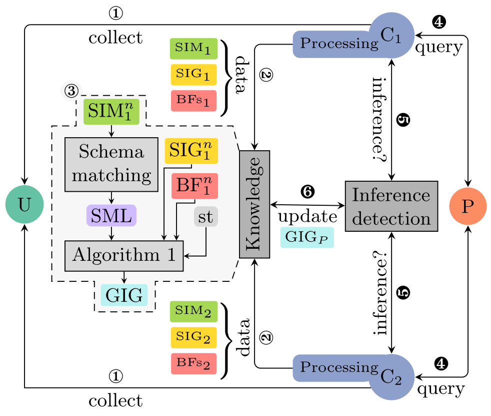
Computing the GIG. Our global graph is initialized by computing the disjoint union of the SIGs issued by the set of data controllers participating to the inference detection solution. Then the GIG computation is completed by linking semantically corresponding nodes related to similar instances from different SIGs with *similarity links*. The naive approach is to first encode instances of each SIG into a BF and proceed to a pairwise similarity score computation. The nodes related to a couple of similar instances are then linked together with a *similarity link* in the GIG. But computing this score for each couple of BFs is expensive. To reduce this cost, we propose to guide the similarity discovery process by beforehand applying schema matching techniques, such as [11], among the SIMs. In fact, a SIM represent dependencies at the schema-level in a database thus by identifying the semantically related attributes in different databases, one can restrict the pairwise similarity score computation to the instances related by a schema matching relation.

Algorithm 1: Computation of the *Global Instance Graph* (GIG)

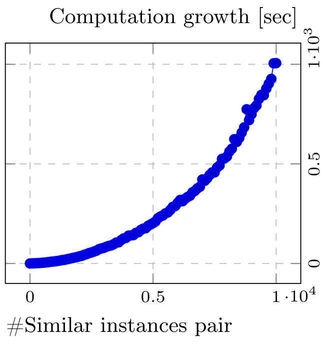
```
Inputs: $\underline{SIG}$: Set of the data controllers' SIG. $\underline{SML}$: Set of schema matching
        relations between the SIMs. $\underline{BF}$: Set of BFs related to instances in the
        SIGs. $\underline{st}$: Similarity threshold.
Outputs: $\underline{M}$: Set of pairs of nodes related to semantically similar instances.
         $\underline{GIG}$: SIGs' linkage based on the instances similarity.
1  Function instance_matching($SIG_i$, $SIG_j$, $SML_{ij}$, $BF_{ij}$, $st$)
2      $M \longleftarrow \emptyset$
3      foreach $l \in SML_{ij}$ do
4          foreach $(n_i, n_j)$ related by $l$, $n_i \in SIG_i$, $n_j \in SIG_j$ do
               // Get the Bloom Filter of a node's instance
5              $bf_i, bf_j \longleftarrow get(BF_{ij}, n_i), get(BF_{ij}, n_j)$
6              if $Dice\_Coefficient(bf_i, bf_j) > st$ then
7                  $M \longleftarrow M \cup (n_i, n_j)$
8      return $M$
9  Function gig_computation(SIG, SML, BF, st)
10     $GIG \longleftarrow$ disjoint union of the SIGs in $SIG$
11     foreach $(SIG_i, SIG_j) \in SIG$, $i < j$, $SML_{ij} \in SML$, $BF_{ij} \in BF$ do
12         $M \longleftarrow instance\_matching(SIG_i, SIG_j, SML_{ij}, BF_{ij}, st)$
13         foreach $(n_k, n_l) \in M$ do
14             $GIG \longleftarrow$ Add a similarity link between $n_k$ and $n_l$ in $GIG$
15     return $GIG$
```

In the following, we present our algorithm for the GIG creation. Algorithm 1 is composed of two functions: (i) *instance_matching* filters pairs of nodes of different SIGs based on the related *schema matching relations* (SML); computes the similarity score of each candidate pair based on *Bloom Filters* (BFs); then collects the pairs of nodes of similar instances based on the similarity threshold, and (ii) *gig_computation* which initialises the GIG with the disjoint union of all the SIGs; processes pairwisely the SIGs to compute the pairs of similar instances; and links nodes related to similar instances with a *similarity link* in the GIG.

The time complexity of Algorithm 1 is related to the number of calls to the $Dice\_Coefficient$ function which computes the similarity score between two BFs (i.e., two instances). In the worst case, for the function *instance_matching* all pairs of nodes $(n_i,\ n_j)$ are related to each semantic matching link l. A similarity score is thus computed for each pair of nodes which leads to a complexity of $O(|SML_{ij}| \cdot |SIG_i| \cdot |SIG_j|)$ where $|.|$ denotes either the cardinality of the set of schema matching relations or the number of instances in a SIG. For readability purposes, the complexity of *instance_matching* is abbreviated as $O(im)$. Then the function *gig_computation* goes through all the combinations of pairs of SIGs without repetition and, in the worst case, calls the function *instance_matching* for each pair. With $|SIG|$ being the number of SIGs, the complexity of *gig_computation* is $O(|SIG|^2 \cdot im)$ since *instance_matching* is called for each combination of SIGs.



(a) Overall workflow of our solution.



(b) GIG computation time growth.

Fig. 2. (a) Overall workflow of our solution. (b) GIG computation time growth.

Workflow. In our architecture, to achieve the privacy preservation of data controller's information, the values of the attributes in the SIGs and the BFs are anonimized by the data controllers themselves so that there is no need to trust the centralized system. As depicted in Fig. 2a, after collecting data from the users ① each data controller must compute its own: SIM, anonymised SIG, and record-level BFs of the instances in the SIG and then sends ② these information to the *knowledge module*. The GIG is built ③ relying on a schema matching technique, such as [11], which processes the set of SIMs from each data controller (i.e., SIM$_1$, ..., SIM$_n$ denoted by SIM$_1^n$). Then Algorithm 1 takes as input the computed schema matching relations, the set of SIGs, the related BFs, and the similarity thresholds in order to compute the *similarity links* between the SIGs and therefore create the GIG. Once computed, for each incoming query from a data processor p ❹, the related data controller sends the anonymized answer of the query to the *inference detection module* ❺. Which, as explained in Sect. 2, relies on the *knowledge module* to query the Bayesian network instance of the GIG assigned to the data processor p issuing the query (i.e., GIG_p) ❻. Once GIG_p is retrieved, the *inference detection module* operates the probabilistic propagation for the *dependency links* as in Chen and manages the *similarity links* as presented in Sect. 3.

4 Experimentation and Validation

In this section we have focused on validating and experimenting the GIG creation step as it represents the novel part of our proposal. The project is hosted at: https://gitlab.com/plht/prototype. Since the system proposed by Chen has not been implemented during this experimentation, several programs have been implemented in order to simulate the *processing* block of the data controllers which build the anonymised SIG and record-level BFs as depicted in Fig. 2a.

Dataset. The requirements that we want for the dataset are: (i) it must contain at least two databases with more than one table to have dependencies between attributes within a table and between different tables (ii) the two schema must have semantically matching attributes (iii) both databases must contain instances that are similar. We did not find a dataset that matches all three requirements. Thus we have adapted the Northix dataset[2] designed for schema matching benchmark in data integration problems. It contains a total of 115 attributes distributed in two databases called *Sakila* and *Northwind*, modeling an online DVD rental store and a fictitious food company respectively. Those databases are often used as samples for learning purposes and experimentation in scientific publications.

The resulting schema matching leads to 110 and 28 relations, between attributes within the same database and between attributes of different databases respectively. In our case, the drawback of this dataset is that it does not contain any similar instances between the two databases. Thus, we have implemented a program which creates either duplicates of customer instances from one database to the other or create pairs of randomly generated similar customer instances in both databases.

Settings & Results. To measure the efficiency of the GIG computation, we choose to focus only on customer instances matching since they represent the type of instances that interests potentially an attacker in a realistic MDIA. The two SIMs[3] used for the experimentation have been created manually. We have been careful to represent semantically realistic dependencies between the

[2] https://archive.ics.uci.edu/ml/datasets/Northix.
[3] https://gitlab.com/plht/prototype/-/tree/master/model#experimentation.

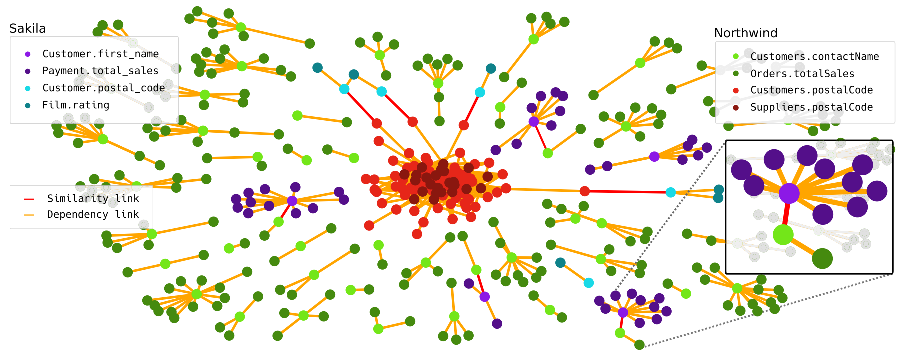


Fig. 3. GIG with four pairs of similar customer instances.

attributes. We have chosen to vary the number of pairs of similar instances from 0 up to 10^4 pairs by inserting 100 new pairs at each step. All points of measure has been repeated 10 times and the median of each repetition is used in the plot. Finally, the measures have been performed in a Docker container hosted on Ubuntu 14.04.6 and running on an Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz with 46 GiB of RAM.

Figure 2b depicts the measure of the GIG computation time growth depending on the number of pairs of similar instances between the two databases of the Northix dataset. The quadratic growth matches the theoretical time complexity of Sect. 3 and is the result of processing pairwisely the SIGs (line 11 in Algorithm 1) in order to compute the matching instances. Nevertheless, the GIG can be computed offline by processing the white steps in Fig. 2a before being used online and the performances when data is queried from the databases are not affected during the black steps.

Figure 3 shows an example of a small GIG with 4 *similarity links* related to the schema matching relation (*postal_code*, *postalCode*) and 4 others related to (*first_name*, *contactName*). In the zoomed sub-graph the two customer instances are linked together by a similarity link. Therefore, when a query is issued to *Sakila* to get the value of *first_name*, then during the probability propagation phase of the Bayesian network instance specific to the user issuing the query, the inference detection algorithm will update the knowledge of *first_name* to 100% since the user now knows the value of this attribute. Next the *similarity link* will be processed by setting the same percentage of knowledge to the node at the other end (i.e., *contactName*). This knowledge propagation through *similarity links* allows the centralised system to detect knowledge queried on one database which can be used to infer sensitive values on other related databases.

The experimental part of our work highlights several issues linked to the use of BF in our solution: first of all the BF limitations in handling heterogeneity among database schemas to identify similarities[4]. On the other hand, the similarity is stated between two instances based on a fixed threshold.

[4] https://gitlab.com/plht/prototype/-/tree/master/dataset#sakila.

5 Conclusion

We have proposed the design of a Multi Database Inference Attack (MDIA) detection system. Our approach extends an existing solution by using schema matching and privacy-preserving record linkage techniques in order to detect inference channels between databases. With respect to our hypothesis, we are able to build a *Global Instance Graph* which represents the inference channels within each (and among) database(s). It allows the detection of MDIA that the usual inference detection systems are not able to identify. This is a first step towards the development of a distributed and fully featured MDIA detection system. In addition, data and schema updates could be integrated in our solution by removing hypothesis (v). In fact, this will affect the models used by the system and requires to propose efficient mechanisms to keep them up-to-date with the concern of maintaining data availability and a good inference detection level.

References

1. Biskup, J.: Dynamic policy adaptation for inference control of queries to a propositional information system. J. Comput. Secur. **20**(5), 509–546 (2012)
2. Brodsky, A., Farkas, C., Jajodia, S.: Secure databases: constraints, inference channels, and monitoring disclosures. IEEE Trans. Knowl. Data Eng. **12**(6), 900–919 (2000)
3. Chen, Y., Chu, W.W.: Database security protection via inference detection. In: Mehrotra, S., Zeng, D.D., Chen, H., Thuraisingham, B., Wang, F.-Y. (eds.) ISI 2006. LNCS, vol. 3975, pp. 452–458. Springer, Heidelberg (2006). https://doi.org/10.1007/11760146_40
4. Christen, P., Ranbaduge, T., Vatsalan, D., Schnell, R.: Precise and fast cryptanalysis for bloom filter based privacy-preserving record linkage. IEEE Trans. Knowl. Data Eng. **31**(11), 2164–2177 (2019)
5. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, vol. 2, pp. 1300–1307. Stockholm Sweden (1999)
6. Gerl, A., Bennani, N., Kosch, H., Brunie, L.: LPL, towards a GDPR-compliant privacy language: formal definition and usage. In: Hameurlain, A., Wagner, R. (eds.) Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVII. LNCS, vol. 10940, pp. 41–80. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-57932-9_2
7. Guarnieri, M., Marinovic, S., Basin, D.: Securing databases from probabilistic inference. In: 2017 IEEE 30th Computer Security Foundations Symposium (CSF), pp. 343–359 (2017)
8. Randall, S.M., Ferrante, A.M., Boyd, J.H., Bauer, J.K., Semmens, J.B.: Privacy-preserving record linkage on large real world datasets. J. Biomed. Inform. **50**, 205–212 (2014)
9. Schnell, R., Borgs, C.: Randomized response and balanced bloom filters for privacy preserving record linkage. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), pp. 218–224 (2016)
10. Staddon, J.: Dynamic inference control. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (2003)

11. Villányi, B., Martinek, P.: DIPROM: DIstance PROportional Matcher exploiting neighbor-levels and related terms. Periodica Polytechnica Electr. Eng. Comput. Sci. **61**(1), 1–11 (2017)
12. Woodall, P., Brereton, P.: A systematic literature review of inference strategies. Int. J. Inf. Comput. Secur. **4**(2), 99–117 (2010)

Author Index