Gesellschaft für Informatik

Proceedings of the 26th GI-Workshop Grundlagen von Datenbanken

Bozen-Bolzano, Italien, 21.-24. Oktober 2014

seit 1558

universität innsbruck

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Herausgeber:

Friederike Klan
Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik
Heinz-Nixdorf-Stiftungsprofessur für Verteilte Informationssysteme
Ernst-Abbe-Platz 2
DE-07743 Jena
E-Mail: *friederike.klan@uni-jena.de*

Günther Specht
Universität Innsbruck
Fakultät für Mathematik, Informatik und Physik
Forschungsgruppe Datenbanken und Informationssysteme
Technikerstrasse 21a
AT-6020 Innsbruck
E-Mail: *guenther.specht@uibk.ac.at*

Hans Gamper
Freie Universität Bozen-Bolzano
Fakultät für Informatik
Dominikanerplatz 3
IT-39100 Bozen-Bolzano
E-Mail: *gamper@inf.unibz.it*

Vorwort

Der 26. Workshop "Grundlagen von Datenbanken" (GvDB) 2014 fand dieses Jahr vom 21.10.2014 bis 24.10.2014 auf dem Ritten in Südtirol statt, einem reizvollen Hochplateau mit Blick auf die Dolomiten. Bereits die Anreise war ein Highlight: vom Bahnhof Bozen ging es mit der längsten Seilbahn Südtirols nach oben und dann mit der Rittner Bahn, einer alten Schmalspur-Strasenbahn, über die Lärchenwiesen bis zum Tagungsort.

Der viertägige Workshop wurde vom GI-Arbeitskreis "Grundlagen von Informationssystemen" im Fachbereich Datenbanken und Informationssysteme (DBIS) veranstaltet und hat die konzeptionellen und methodischen Grundlagen von Datenbanken und Informationssystemen zum Thema, ist aber auch für neue Anwendungen offen. Die Workshopreihe und der Arbeitskreis feiern dieses Jahr ihr 25-jähriges Bestehen. Der AK ist damit der ältesten Arbeitskreise der GI. Organisiert wurde der Jubiläumsworkshop gemeinsam von Fr. Dr. Friederike Klan von der Heinz-Nixdorf-Stiftungsprofessur für Verteilte Informationssysteme der Friedrich-Schiller-Universität Jena, Hr. Prof. Dr. Günther Specht von der Forschungsgruppe Datenbanken und Informationssysteme (DBIS) der Universität Innsbruck und Hr. Prof. Dr. Johann Gamper von der Gruppe Datenbanken und Informationssysteme (DIS) der Freien Universität Bozen-Bolzano.

Der Workshop soll die Kommunikation zwischen Wissenschaftlern/-innen im deutschsprachigen Raum fördern, die sich grundlagenorientiert mit Datenbanken und Informationssystemen beschäftigen. Er bietet insbesondere Nachwuchswissen-schaftler/-innen die Möglichkeit, ihre aktuellen Arbeiten einem gröseren Forum in lockerer Atmosphäre vorzustellen. Mit der Kulisse der beeindruckenden Südtiroler Bergwelt bot der Workshop auf 1200 Metern Meereshöhe einen idealen Rahmen für offene und inspirierende Diskussionen dazu ohne Zeitzwang. Insgesamt wurden 14 Arbeiten aus den Einsendungen nach einem Review-Prozess ausgewählt und vorgestellt. Besonders hervorzuheben ist die Vielfältigkeit der Themenbereiche: sowohl Kerngebiete in Datenbanksystemen bzw. Datenbankdesign, als auch Themen zur Informationsextraktion, Empfehlungssysteme, Verarbeitung von Zeitreihen, Graphalgorithmen im GIS Bereich, sowie zu Datenschutz und Datenqualität wurden vorgestellt.

Die Vorträge ergänzten zwei Keynotes: Ulf Leser, Professor an der Humboldt-Universität zu Berlin, hielt eine Keynote zu Next Generation Data Integration (for Life Sciences) und Francesco Ricci, Professor an der Freien Universität von Bozen-Bolzano zu Context and Recommendations: Challenges and Results. Beiden Vortragenden sei an dieser Stelle für ihre spontane Bereitschaft zu Kommen und ihre interessanten Vorträge gedankt.

Neben dem Wissensaustausch darf auch die soziale Komponente nicht fehlen. Die beiden gemeinsamen Ausflüge bleiben sicher allen lange in guter Erinnerung. Zum einen erklommen wir das bereits schneebedeckte Rittner Horn (2260 Hm), von dem man einen herrlichen Blick auf die Dolomiten hat. Zum anderen ist bei einem Aufenthalt im Herbst in Südtirol das so genannte Törggelen nicht wegzudenken: eine Wanderung zu lokalen Bauernschänken, die die Köstlichkeiten des Jahres zusammen mit Kastanien und heurigem Wein auftischen. Sogar der Rektor der Universität Bozen-Bolzano kam dazu extra vom Tal herauf.

Eine Konferenz kann nur erfolgreich in einer guten Umgebung stattfinden. Daher danken wir an dieser Stelle den Mitarbeitern des Hauses der Familie für ihre Arbeit im Hintergrund. Weiterer Dank gilt allen Autoren, die mit ihren Beiträgen und Vorträgen erst einen interessanten Workshop ermöglichen, sowie dem Programmkomitee und allen Gutachtern für ihre Arbeit. Abschlie̋send gilt dem Organisationsteam, das interaktiv über alle Landesgrenzen hinweg (Deutschland, Österreich und Italien) hervorragend zusammen gearbeitet hat, ein grőes Dankeschön. So international war der GvDB noch nie.

Auf ein Wiedersehen beim nächsten GvDB-Workshop

Günther Specht
Friederike Klan
Johann Gamper

Innsbruck, Jena, Bozen am 26.10.2014

Komitee

Organisation

Friederike Klan	Friedrich-Schiller-Universität Jena
Günther Specht	Universität Innsbruck
Hans Gamper	Universität Bozen-Bolzano

Programm-Komitee

Alsayed Algergawy	Friedrich-Schiller-Universität Jena
Erik Buchmann	Karlsruher Institut für Technologie
Stefan Conrad	Universität Düsseldorf
Hans Gamper	Universität Bozen-Bolzano
Torsten Grust	Universität Tübingen
Andreas Heuer	Universität Rostock
Friederike Klan	Friedrich-Schiller-Universität Jena
Birgitta König-Ries	Friedrich-Schiller-Universität Jena
Klaus Meyer-Wegener	Universität Erlangen
Gunter Saake	Universität Magdeburg
Kai-Uwe Sattler	Technische Universität Ilmenau
Eike Schallehn	Universität Magdeburg
Ingo Schmitt	Brandenburgische Technische Universität Cottbus
Holger Schwarz	Universität Stuttgart
Günther Specht	Universität Innsbruck

Zusätzliche Reviewer

Mustafa Al-Hajjaji	Universität Magdeburg
Xiao Chen	Universität Magdeburg
Doris Silbernagl	Universität Innsbruck

Contents

Next Generation Data Integration (for the Life Sciences)

[Abstract]

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

ABSTRACT

Ever since the advent of high-throughput biology (e.g., the Human Genome Project), integrating the large number of diverse biological data sets has been considered as one of the most important tasks for advancement in the biological sciences. The life sciences also served as a blueprint for complex integration tasks in the CS community, due to the availability of a large number of highly heterogeneous sources and the urgent integration needs. Whereas the early days of research in this area were dominated by virtual integration, the currently most successful architecture uses materialization. Systems are built using ad-hoc techniques and a large amount of scripting. However, recent years have seen a shift in the understanding of what a "data integration system" actually should do, revitalizing research in this direction. In this tutorial, we review the past and current state of data integration (exemplified by the Life Sciences) and discuss recent trends in detail, which all pose challenges for the database community.

About the Author

Ulf Leser obtained a Diploma in Computer Science at the Technische Universität München in 1995. He then worked as database developer at the Max-Planck-Institute for Molecular Genetics before starting his PhD with the Graduate School for "Distributed Information Systems" in Berlin. Since 2002 he is a professor for Knowledge Management in Bioinformatics at Humboldt-Universität zu Berlin.

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

Context and Recommendations: Challenges and Results

[Abstract]

Francesco Ricci
Free University of Bozen-Bolzano
Faculty of Computer Science
fricci@unibz.it

ABSTRACT

Recommender Systems (RSs) are popular tools that automatically compute suggestions for items that are predicted to be interesting and useful to a user. They track users' actions, which signal users' preferences, and aggregate them into predictive models of the users' interests. In addition to the long-term interests, which are normally acquired and modeled in RSs, the specific ephemeral needs of the users, their decision biases, the context of the search, and the context of items' usage, do influence the user's response to and evaluation for the suggested items. But appropriately modeling the user in the situational context and reasoning upon that is still challenging; there are still major technical and practical difficulties to solve: obtaining sufficient and informative data describing user preferences in context; understanding the impact of the contextual dimensions on user decision-making process; embedding the contextual dimensions in a recommendation computational model. These topics will be illustrated in the talk, making examples taken from the recommender systems that we have developed.

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

About the Author

Francesco Ricci is associate professor of computer science at Free University of Bozen-Bolzano, Italy. His current research interests include recommender systems, intelligent interfaces, mobile systems, machine learning, case-based reasoning, and the applications of ICT to tourism and eHealth. He has published more than one hundred of academic papers on these topics and has been invited to give talks in many international conferences, universities and companies. He is among the editors of the Handbook of Recommender Systems (Springer 2011), a reference text for researchers and practitioners working in this area. He is the editor in chief of the Journal of Information Technology & Tourism and in the editorial board of the Journal of User Modeling and User Adapted Interaction. He is member of the steering committee of the ACM Conference on Recommender Systems. He served on the program committees of several conferences, including as a program co-chair of the ACM Conference on Recommender Systems (RecSys), the International Conference on Case-Based Reasoning (ICCBR) and the International Conference on Information and Communication Technologies in Tourism (ENTER).

Optimization of Sequences of XML Schema Modifications - The ROfEL Approach

Thomas Nösinger, Meike Klettke, Andreas Heuer
Database Research Group
University of Rostock, Germany
(tn, meike, ah)@informatik.uni-rostock.de

ABSTRACT

The transformation language ELaX (Evolution Language for XML-Schema [16]) is a domain-specific language for modifying existing XML Schemas. ELaX was developed to express complex modifications by using add, delete and update statements. Additionally, it is used to consistently log all change operations specified by a user. In this paper we present the rule-based optimization algorithm ROfEL (Rule-based Optimizer for ELaX) for reducing the number of logged operations by identifying and removing unnecessary, redundant and also invalid modifications. This is an essential prerequisite for the co-evolution of XML Schemas and corresponding XML documents.

1. INTRODUCTION

The eXtensible Markup Language (XML) [2] is one of the most popular formats for exchanging and storing structured and semi-structured information in heterogeneous environments. To assure that well-defined XML documents are valid it is necessary to introduce a document description, which contains information about allowed structures, constraints, data types and so on. XML Schema [4] is one commonly used standard for dealing with this problem. After using an XML Schema a period of time, the requirements can change; for example if additional elements are needed, data types change or integrity constraints are introduced. This may result in the adaptation of the XML Schema definition.

In [16] we presented the transformation language ELaX (Evolution Language for XML-Schema) to describe and formulate these XML Schema modifications. Furthermore, we mentioned briefly that ELaX is also useful to log information about modifications consistently, an essential prerequisite for the co-evolution process of XML Schema and corresponding XML documents [14].

One problem of storing information over a long period of time is, that there can be different unnecessary or redundant modifications. Consider modifications which firstly add an element and shortly afterwards delete the same element. In the overall context of an efficient realization of modification steps, such operations have to be removed. Further issues are incorrect information (possibly caused by network problems), for example if the same element is deleted twice or the order of modifications is invalid (e.g. update before add).

The new rule-based optimizer for ELaX (ROfEL - Rule-based Optimizer for ELaX) had been developed for solving the above mentioned problems. With ROfEL it is possible to identify unnecessary or redundant operations by using different straightforward optimization rules. Furthermore, the underlying algorithm is capable to correct invalid modification steps. All in all, ROfEL could reduce the number of modification steps by removing or even correcting the logged ELaX operations.

This paper is organized as follows. **Section 2** gives the necessary background of XML Schema, ELaX and corresponding concepts. **Section 3** and **section 4** present our approach, by first specifying our ruled-based algorithm ROfEL and then showing how our approach can be applied for an example. Related work is shown in **section 5**. Finally, in **section 6** we draw our conclusions.

2. TECHNICAL BACKGROUND

In this section we present a common notation used in the remainder of this paper. At first, we will shortly introduce the XSD (XML Schema Definition [4]), before details concerning ELaX (Evolution Language for XML-Schema [16]) and the logging of ELaX are given.

The XML Schema *abstract data model* consists of different components (simple and complex type definitions, element and attribute declarations, etc.). Additionally, the *element information item* serves as an XML representation of these components and defines which content and attributes can be used in an XML Schema. The possibility of specifying declarations and definitions in a local or global scope leads to four different modeling styles [13]. One of them is the *Garden of Eden* style, in which all above mentioned components are globally defined. This results in a high re-usability of declarations and defined data types and influences the flexibility of an XML Schema in general.

The transformation language ELaX[1] was developed to handle modifications on an XML Schema and to express such modifications formally. The *abstract data model*, *element information item* and *Garden of Eden* style were important through the development process and influence

[1]The whole transformation language ELaX is available at: `www.ls-dbis.de/elax`

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

the EBNF (Extended Backus-Naur Form) like notation of ELaX.

An ELaX statement always starts with "add", "delete" or "update" followed by one of the alternative components (simple type, element declaration, etc.), an identifier of the current component and completed with optional tuples of attributes and values (examples follow on, e.g. see figure 1). The identifier is a unique *EID* (emxid)[2], a QNAME (qualified name) or a subset of XPath expressions. In the remaining parts we will use the *EID* as the identifier, but a transformation would be easily possible.

ELaX statements are logged for further analyses and also as a prerequisite for the rule-base optimizer (see section 3). Figure 1 illustrates the relational schema of the log. The

file-ID	time	EID	op-Type	msg-Type	content
1	1	1	add	0	add element name 'name' type 'xs:decimal' id 'EID1' ;
1	2	1	upd	0	update element name 'name' change type 'xs:string' ;
1	3	2	add	0	add element name 'count' type 'xs:decimal' id 'EID2' ;
...	...	...	...	...	...

Figure 1: Schema with relation for logging ELaX

chosen values are simple ones (especially the length). The attributes *file-ID* and *time* are the composite key for the logging relation, the *EID* represents the unique identifier for a component of the XSD. The *op-Type* is a short form for add, delete (del) or update (upd) operations, the *msg-Type* is for the different message types (ELaX (0), etc.). Lastly, the *content* contains the logged ELaX statements. The *file-ID* and *msg-Type* are management information, which are not covered in this paper.

3. RULE-BASED OPTIMIZER

The algorithm ROfEL (Rule-based Optimizer for ELaX) was developed to reduce the number of logged ELaX operations. This is possible by combining given operations and/or removing unnecessary or even redundant operations. Furthermore, the algorithm could identify invalid operations in a given log and correct these to a certain degree.

ROfEL is a rule-based algorithm. Provided that a log of ELaX operations is given (see section 2), the following rules are essential to reduce the number of operations. In compliance with ELaX these operations are delete (**del**), **add** or update (**upd**). If a certain distinction is not necessary a general operation (**op**) or variable (_) are used, **empty** denotes a not given operation. Additionally, the rules are classified by their purpose to handle redundant (**R**), unnecessary (**U**) or invalid (**I**) operations. ROfEL stops (**S**) if no other rules are applicable, for example no other operation with the same EID is given.

$$\text{S: } \mathit{empty} \rightarrow \boldsymbol{op(EID)} \Rightarrow op(EID) \tag{1}$$

// ↓ most recent operation: delete (**del**) ↓

$$\text{R: } del(EID) \rightarrow \boldsymbol{del(EID)} \Rightarrow del(EID) \tag{2}$$

$$\text{U: } add(EID, content) \rightarrow \boldsymbol{del(EID)} \Rightarrow \mathit{empty} \tag{3}$$

$$\text{U: } upd(EID, content) \rightarrow \boldsymbol{del(EID)} \Rightarrow del(EID) \tag{4}$$

with *time(del(EID)) :=* **TIME***(del(EID), upd(EID, content))*

[2]Our conceptual model is EMX (Entity Model for XML Schema [15]), in which every component of a model has its own, global identifier: *EID*

// ↓ most recent operation: **add** ↓

$$\text{U: } op(EID) \rightarrow del(EID) \rightarrow \boldsymbol{add(EID,\ content)} \Rightarrow op(EID) \rightarrow \boldsymbol{add(EID,\ content)} \tag{5}$$

$$\text{I: } add(EID, \_) \rightarrow \boldsymbol{add(EID,\ content)} \Rightarrow add(EID, content) \tag{6}$$

$$\text{I: } upd(EID, \_) \rightarrow \boldsymbol{add(EID,\ content)} \Rightarrow upd(EID, content) \tag{7}$$

// ↓ most recent operation: update (**upd**) ↓

$$\text{I: } op(EID) \rightarrow del(EID) \rightarrow \boldsymbol{upd(EID,\ content)} \Rightarrow op(EID) \rightarrow \boldsymbol{upd(EID,\ content)} \tag{8}$$

$$\text{U: } add(EID, content) \rightarrow \boldsymbol{upd(EID,\ content)} \Rightarrow add(EID, content) \tag{9}$$

$$\text{U: } add(EID, content) \rightarrow \boldsymbol{upd(EID,\ content')} \Rightarrow add(EID, \mathbf{MERGE}(content', content)) \tag{10}$$

$$\text{R: } upd(EID, content) \rightarrow \boldsymbol{upd(EID,\ content)} \Rightarrow upd(EID, content) \tag{11}$$

$$\text{U: } upd(EID, content) \rightarrow \boldsymbol{upd(EID,\ content')} \Rightarrow upd(EID, \mathbf{MERGE}(content', content)) \tag{12}$$

The rules have to be sequentially analyzed from left to right (→), whereas the left operation comes temporally before the right one (i.e., time(left) < time(right). To warrant that the operations are working on the same component, the EID of both operations is equal. If two operations exist and a rule applies to them, then the result can be found on the right side of ⇒. The time of the result is the time of the prior (left) operation, except further investigations are explicit necessary or the time is unknown (e.g. empty).

Another point of view illustrates, that the introduced rules are complete concerning the given operations add, delete and update. Figure 2 represents an operation matrix, in which every possible combination is covered with at least one rule. On the x-axis the prior operation and on the y-axis the

operation		prior		
		add	delete	update
recent	add	(6)	(5)	(7)
	delete	(3)	(2)	(4)
	update	(9) , (10)	(8)	(11) , (12)

Figure 2: Operation matrix of rules

recent operation are given, whereas the three-valued rules (5) and (8) are minimized to the both most recent operations (e.g. without *op(EID)*). The break-even point contains the applying rule or rules (considering the possibility of merging the content, see below).

Rule (4) is one example for further investigations. If a component is deleted (*del(EID)*) but updated (*upd(EID)*) before, then it is not possible to replace the prior operation with the result (*del(EID)*) without analyzing other operations between them. The problem is: if another operation (*op(EID')*) references the deleted component (e.g. a simple type) but because of ROfEL *upd(EID)* (it is the prior operation) is replaced with *del(EID)*, then *op(EID')* would be invalid. Therefore, the function **TIME()** is used to determine the correct time of the result. The function is given in pseudocode in figure 3. **TIME()** has two input parame-

```
TIME(op, op'):
// time(op) = t; time(op') = t'; time(opx) = tx;
// op.EID == op'.EID; op.EID != opx.EID; t > t';
   begin
     if ((t > tx > t') AND
         (op.EID in opx.content))
       then return t;
     return t';
   end.
```

Figure 3: TIME() function of optimizer

```
MERGE(content, content'):
// content = (A1 = 'a1', A2 = 'a2',
//            A3 = '', A4 = 'a4');
// content' = (A1 = 'a1', A2 = '',
//            A3 = 'a3', A5 = 'a5');
   begin
     result := {};
     count := 1;
     while (count <= content.size())
       result.add(content.get(count));
       if (content.get(count) in content')
         then
           content'.remove(content.get(count));
       count := count + 1;
     count := 1;
     while (count <= content'.size())
       result.add(content'.get(count));
       count := count + 1;
// result = (A1 = 'a1', A2 = 'a2',
//           A3 = '', A4 = 'a4', A5 = 'a5');
     return result;
   end.
```

Figure 4: MERGE() function of optimizer

ters and returns a time value, dependent on the existence of an operation, which references the EID in its content. If no such operation exists, the time of the result in rule (4) would be the time of the left (*op*), otherwise of the right operation (*op'*). The lines with // are comments and contain further information, some hints or even explanations of variables.

The rules (6), (7) and (8) adapt invalid operations. For example if a component is updated but deleted before (see rule (8)), then ROfEL has to decide, which operation is valid. In this and similar cases the most recent operation is preferred, because it is more difficult (or even impossible) to check the intention of the prior operation. Consequently, in rule (8) *del(EID)* is removed and rule *op(EID)* $\rightarrow$ *upd(EID, content)* applies (*op(EID)* could be *empty*; see rule (1)).

The rules (10) and (12) removes unnecessary operations by merging the content of the involved operations. The function **MERGE()** implements this, the pseudocode is presented in figure 4. **MERGE()** has two input parameter, the content of the most recent (left) and prior (right) operation. The content is given as a sequence of attribute-value pairs (see ELaX description in section 2). The result of the function is the combination of the input, whereas the content of the most recent operation is preferred analogical to the above mentioned behaviour for **I** rules. All attribute-value pairs of the most recent operation are completely inserted into the result. Simultaneously, these attributes are removed from the content of the prior operation. At the end of the function, all remaining attributes of the prior (right) operation are inserted, before the result is returned.

All mentioned rules, as well as the functions **TIME()** and **MERGE()** are essential parts of the main function **ROFEL()**; the pseudocode is presented in figure 5. **ROFEL()**

```
ROFEL(log):
// log = ((t1,op1), (t2,op2), ...); t1 < t2 < ...;
 begin
   for (i := log.size(); i >= 2; i := i - 1)
     for (k := i - 1; k >= 1 ; k := k - 1)
       if(!(log.get(i).EID == log.get(k).EID AND
           log.get(i).time != log.get(k).time))
         then continue;
// R: del(EID) -> del(EID) => del(EID) (2)
       if (log.get(i).op-Type == 1 AND
           log.get(k).op-Type == 1)
         then
           log.remove(i);
           return ROFEL(log);
// U: upd(EID, content) -> del(EID)
//     => del(EID) (4)
       if (log.get(i).op-Type == 1 AND
           log.get(k).op-Type == 2)
         then
           temp := TIME(log.get(i), log.get(k));
           if (temp == log.get(i).time)
             then
               log.remove(k);
               return ROFEL(log);
           log.get(k) := log.get(i);
           log.remove(i);
           return ROFEL(log); [...]
// U: upd(EID,con) -> upd(EID,con')
//     => upd(EID, MERGE(con',con)) (12)
       if (log.get(i).op-Type == 2 AND
           log.get(k).op-Type == 2)
         then
           temp := MERGE(log.get(i).content,
                         log.get(k).content);
           log.get(k).content := temp;
           log.remove(i);
           return ROFEL(log);
   return log;
 end.
```

Figure 5: Main function ROFEL() of optimizer

has one input parameter, the log of ELaX operations. This log is a sequence sorted according to time, it is analyzed reversely. In general, one operation is pinned (*log.get(i)*) and compared with the next, prior operation (*log.get(k)*). If *log.get(k)* modifies the same component as *log.get(i)* (i.e., EID is equal) and the time is different, then an applying rule is searched, otherwise the next operation (*log.get(k - 1)*) is analyzed. The algorithm terminates, if the outer loop completes successfully (i.e., no further optimization is possible).

Three rules are presented in figure 5; the missing ones are skipped (**[...]**). The first rule is (2), the occurrence of redundant delete operations. According to the above mentioned time choosing guidelines, the most recent operation (*log.get(i)*) is removed. After this the optimizer starts again with the modified log recursively (*return ROFEL(log)*).

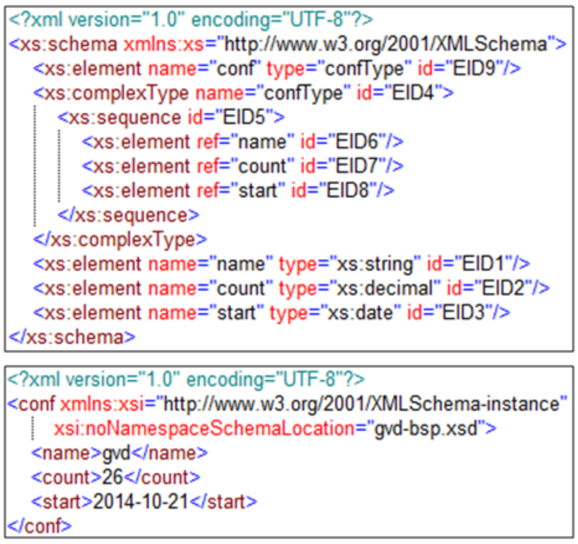
The second rule is (4), which removes an unnecessary update operation, because the whole referenced component will be deleted later. This rule uses the **TIME()** function of figure 3 to decide, which time should be assigned to the result. If another operation between *log.get(i)* and *log.get(k)* exists and this operation contains or references *log.get(i).EID*, then the most recent time (*log.get(i).time*) is assigned, otherwise the prior time (*log.get(k).time*).

The last rule is (12), different updates on the same component are given. The **MERGE()** function of figure 4 combines the content of both operations, before the content of the prior operation is changed and the most recent operation is removed.

After introducing detailed information about the concept of the ROfEL algorithm, we want to use it to optimize an example in the next section.

4. EXAMPLE

In the last section we specified the rule-based algorithm ROfEL (Rule-based Optimizer for ELaX), now we want to explain the use with an example: we want to store some information about a conference. We assume the XML Schema of figure 6 is given, a corresponding XML document is also presented. The XML Schema is in the *Garden of Eden* style



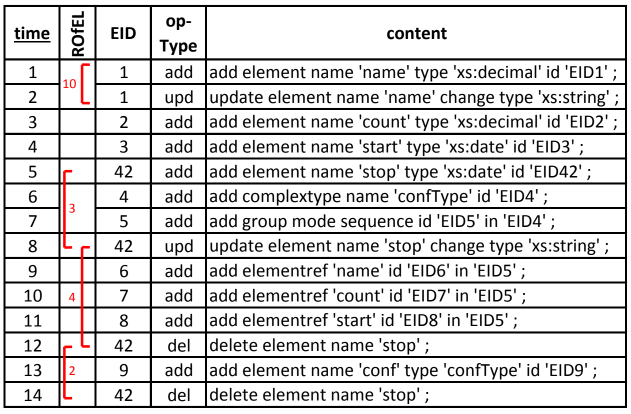
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="conf" type="confType" id="EID9"/>
  <xs:complexType name="confType" id="EID4">
    <xs:sequence id="EID5">
      <xs:element ref="name" id="EID6"/>
      <xs:element ref="count" id="EID7"/>
      <xs:element ref="start" id="EID8"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="name" type="xs:string" id="EID1"/>
  <xs:element name="count" type="xs:decimal" id="EID2"/>
  <xs:element name="start" type="xs:date" id="EID3"/>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<conf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="gvd-bsp.xsd">
  <name>gvd</name>
  <count>26</count>
  <start>2014-10-21</start>
</conf>
```

Figure 6: XML Schema with XML document

and contains four element declarations (*conf*, *name*, *count*, *start*) and one complex type definition (*confType*) with a group model (*sequence*). The group model has three element references, which reference one of the simple type element declarations mentioned above. The identification of all components is simplified by using an EID, it is visualized as a unique ID attribute (*id = ”..”*).

The log of modification steps to create this XML Schema is presented in figure 7. The relational schema is reduced in comparison to figure 1. The *time*, the component *EID*, the *op-Type* and the *content* of the modification steps are given. The log contains different modification steps, which are not



time	ROfEL	EID	op-Type	content
1	10	1	add	add element name 'name' type 'xs:decimal' id 'EID1' ;
2		1	upd	update element name 'name' change type 'xs:string' ;
3		2	add	add element name 'count' type 'xs:decimal' id 'EID2' ;
4		3	add	add element name 'start' type 'xs:date' id 'EID3' ;
5		42	add	add element name 'stop' type 'xs:date' id 'EID42' ;
6	3	4	add	add complextype name 'confType' id 'EID4' ;
7		5	add	add group mode sequence id 'EID5' in 'EID4' ;
8		42	upd	update element name 'stop' change type 'xs:string' ;
9		6	add	add elementref 'name' id 'EID6' in 'EID5' ;
10	4	7	add	add elementref 'count' id 'EID7' in 'EID5' ;
11		8	add	add elementref 'start' id 'EID8' in 'EID5' ;
12		42	del	delete element name 'stop' ;
13	2	9	add	add element name 'conf' type 'confType' id 'EID9' ;
14		42	del	delete element name 'stop' ;

Figure 7: XML Schema modification log of figure 6

given in the XML Schema (EID > 9). Additionally, some entries are connected within the new introduced column *ROfEL*. The *red lines and numbers* represent the involved log entries and applying ROfEL rule.

The sorted log is analyzed reversely, the operation with time stamp 14 is pinned and compared with time entry 13. Because the modified component is not the same (EID not equal), the next operation with time 12 is taken. Both operations delete the same component (*op-Type == 1*). According to rule (2), the redundant entry 14 is removed and ROFEL restarts with the adapted log.

Rule (4) applies next, a component is updated but deleted later. This rule calls the **TIME()** function to determine, if the time of the result (i.e., *del(EID)*) should be 12 or 8. Because no operation between 12 and 8 references EID 42, the time of the result of (4) is 8. The *content* of time 8 is replaced with *delete element name 'stop';*, the *op-Type* is set to 1 and the time entry 12 is deleted.

Afterwards, ROFEL restarts again and rule (3) could be used to compare the new operation of entry 8 (original entry 12) with the operation of time 5. A component is inserted but deleted later, so all modifications on this component are unnecessary in general. Consequently, both entries are deleted and the component with EID 42 is not given in the XML Schema of figure 6.

The last applying rule is (10). An element declaration is inserted (time 1) and updated (time 2). Consequently, the **MERGE()** function is used to combine the content of both operations. According to the ELaX specification, the content of the update operation contains the attribute *type* with the value *xs:string*, whereas the add operation contains the attribute *type* with the value *xs:decimal* and *id* with *EID1*. All attribute-value pairs of the update operation are completely inserted into the output of the function (*type = ”xs:string”*). Simultaneously, the attribute *type* is removed from the content of the add operation (*type = ”xs:decimal”*). The remaining attributes are inserted in the output (*id = ”EID1”*). Afterwards, the content of entry 1 is replaced by *add element 'name' type ”xs:string” id ”EID1”;* and the second entry is deleted (time 2).

The modification log of figure 7 is optimized with rules (2), (4), (3) and (10). It is presented in figure 8. All in all, five of 14 entries are removed, whereas one is replaced by a combination of two others.

time	EID	op-Type	content
1	1	add	add element name 'name' type 'xs:string' id 'EID1' ;
3	2	add	add element name 'count' type 'xs:decimal' id 'EID2' ;
4	3	add	add element name 'start' type 'xs:date' id 'EID3' ;
6	4	add	add complextype name 'confType' id 'EID4' ;
7	5	add	add group mode sequence id 'EID5' in 'EID4' ;
9	6	add	add elementref 'name' id 'EID6' in 'EID5' ;
10	7	add	add elementref 'count' id 'EID7' in 'EID5' ;
11	8	add	add elementref 'start' id 'EID8' in 'EID5' ;
13	9	add	add element name 'conf' type 'confType' id 'EID9' ;

Figure 8: XML Schema modification log of figure 7 after using rules (2), (4), (3) and (10) of ROfEL

This simple example illustrates how ROfEL can reduce the number of logged operations introduced in section 3. More complex examples are easy to construct and can be solved by using the same rules and the same algorithm.

5. RELATED WORK

Comparable to the object lifecycle, we create new types or elements, use (e.g. modify, move or rename) and delete them. The common optimization rules to reduce the number of operations are originally introduced in [10] and are available in other application in the same way. In [11], rules for reducing a list of user actions (e.g. move, replace, delete, ...) are introduced. In [9], pre and postconditions of operations are used for deciding which optimizations can be executed. Additional applications can easily be found in further scientific disquisitions.

Regarding other transformation languages, the most commonly used are XQuery [3] and XSLT (Extensible Stylesheet Language Transformations [1]), there are also approaches to reduce the number of unnecessary or redundant operations. Moreover, different transformations to improve efficiency are mentioned.

In [12] different "high-level transformations to prune and merge the stream data flow graph" [12] are applied. "Such techniques not only simplify the later analyses, but most importantly, they can rewrite some queries" [12], an essential prerequisite for the efficient evaluation of XQuery over streaming data.

In [5] packages are introduced because of efficiency benefits. A package is a collection of stylesheet modules "to avoid compiling libraries repeatedly when they are used in multiple stylesheets, and to avoid holding multiple copies of the same library in memory simultaneously" [5]. Furthermore, XSLT works with templates and matching rules for identifying structures in general. If different templates could be applied, automatic or user given priorities manage which template is chosen. To avoid unexpected behaviour and improve the efficiency of analyses, it is a good practise to remove unnecessary or redundant templates.

Another XML Schema modification language is XSchemaUpdate [6], which is used in the co-evolution prototype EXup [7]. Especially the auto adaptation guidelines are similar to the ROfEL purpose of reducing the number of modification steps. "Automatic adaptation will insert or remove the minimum allowed number of elements for instance" [6], i.e., "a minimal set of updates will be applied to the documents" [6].

In [8] an approach is presented, which deals with four operations (insert, delete, update, move) on a tree representation of XML. It is similar to our algorithm, but we use ELaX as basis and EIDs instead of update-intensive labelling mechanisms. Moreover the distinction between property and node, the "deletion always wins" view, as well as the limitation that "reduced sequence might still be reducible" [8] are drawbacks. The optimized reduction algorithm eliminates the last drawback, but needs another complex structure, an operation hyper-graph.

6. CONCLUSION

The rule-based algorithm ROfEL (Rule-based Optimizer for ELaX) was developed to reduce the number of logged ELaX (Evolution Language for XML-Schema [16]) operations. In general ELaX statements are add, delete and update operations on the components of XML Schema, specified by a user.

ROfEL allows the identification and deletion of unnecessary and redundant modifications by applying different heuristic rules. Additionally, invalid operations are also corrected or removed. In general if the preconditions and conditions for an adaptation of two ELaX log entries are satisfied (e.g. EID equivalent, op-Type correct, etc.), one rule is applied and the modified, reduced log is returned.

We are confident, that even if ROfEL is domain specific and the underlying log is specialized for our needs, the above specified rules are applicable in other scenarios or applications, in which the common modification operations add, delete and update are used (minor adaptations preconditioned).

Future work. The integration of a cost-based component in ROfEL could be very interesting. It is possible, that under consideration of further analyses the combination of different operations (e.g. rule (10)) is inefficient in general. In this and similar cases a cost function with different thresholds could be defined to guarantee, that only efficient adaptations of the log are applied. A convenient cost model would be necessary, but this requires further research.

Feasibility of the approach. At the University of Rostock we implemented the prototype CodeX (Conceptual design and evolution for XML Schema) for dealing with the co-evolution [14] of XML Schema and XML documents; ROfEL and corresponding concepts are fully integrated. As we plan to report in combination with the first release of CodeX, the significantly reduced number of logged operations proves that the whole algorithm is definitely feasible.

7. REFERENCES

[1] XSL Transformations (XSLT) Version 2.0. `http://www.w3.org/TR/2007/REC-xslt20-20070123/`, January 2007. Online; accessed 25-June-2014.

[2] Extensible Markup Language (XML) 1.0 (Fifth Edition). `http://www.w3.org/TR/2008/REC-xml-20081126/`, November 2008. Online; accessed 25-June-2014.

[3] XQuery 1.0: An XML Query Language (Second Edition). `http://www.w3.org/TR/2010/REC-xquery-20101214/`, December 2010. Online; accessed 25-June-2014.

[4] W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. `http://www.w3.org/TR/2012/`

`REC-xmlschema11-1-20120405/`, April 2012. Online; accessed 25-June-2014.

[5] XSL Transformations (XSLT) Version 3.0. `http://www.w3.org/TR/2013/WD-xslt-30-20131212/`, December 2013. Online; accessed 25-June-2014.

[6] F. Cavalieri. Querying and Evolution of XML Schemas and Related Documents. Master's thesis, University of Genova, 2009.

[7] F. Cavalieri. EXup: an engine for the evolution of XML schemas and associated documents. In *Proceedings of the 2010 EDBT/ICDT Workshops*, EDBT '10, pages 21:1–21:10, New York, NY, USA, 2010. ACM.

[8] F. Cavalieri, G. Guerrini, M. Mesiti, and B. Oliboni. On the Reduction of Sequences of XML Document and Schema Update Operations. In *ICDE Workshops*, pages 77–86, 2011.

[9] H. U. Hoppe. Task-oriented Parsing - a Diagnostic Method to Be Used Adaptive Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, pages 241–247, New York, NY, USA, 1988. ACM.

[10] M. Klettke. *Modellierung, Bewertung und Evolution von XML-Dokumentkollektionen.* Habilitation, Fakultät für Informatik und Elektrotechnik, Universität Rostock, 2007.

[11] R. Kramer. iContract - the Java(tm) Design by Contract(tm) tool. In *In TOOLS '98: Proceedings of the Technology of Object-Oriented Languages and Systems*, page 295. IEEE Computer Society, 1998.

[12] X. Li and G. Agrawal. Efficient Evaluation of XQuery over Streaming Data. In *In Proc. VLDB'05*, pages 265–276, 2005.

[13] E. Maler. Schema Design Rules for UBL...and Maybe for You. In *XML 2002 Proceedings by deepX*, 2002.

[14] T. Nösinger, M. Klettke, and A. Heuer. Evolution von XML-Schemata auf konzeptioneller Ebene - Übersicht: Der CodeX-Ansatz zur Lösung des Gültigkeitsproblems. In *Grundlagen von Datenbanken*, pages 29–34, 2012.

[15] T. Nösinger, M. Klettke, and A. Heuer. A Conceptual Model for the XML Schema Evolution - Overview: Storing, Base-Model-Mapping and Visualization. In *Grundlagen von Datenbanken*, 2013.

[16] T. Nösinger, M. Klettke, and A. Heuer. XML Schema Transformations - The ELaX Approach. In *DEXA (1)*, pages 293–302, 2013.

Automatic Decomposition of Multi-Author Documents Using Grammar Analysis

Michael Tschuggnall and Günther Specht
Databases and Information Systems
Institute of Computer Science, University of Innsbruck, Austria
{michael.tschuggnall, guenther.specht}@uibk.ac.at

ABSTRACT

The task of text segmentation is to automatically split a text document into individual subparts, which differ according to specific measures. In this paper, an approach is presented that attempts to separate text sections of a collaboratively written document based on the grammar syntax of authors. The main idea is thereby to quantify differences of the grammatical writing style of authors and to use this information to build paragraph clusters, whereby each cluster is assigned to a different author. In order to analyze the style of a writer, text is split into single sentences, and for each sentence a full parse tree is calculated. Using the latter, a profile is computed subsequently that represents the main characteristics for each paragraph. Finally, the profiles serve as input for common clustering algorithms. An extensive evaluation using different English data sets reveals promising results, whereby a supplementary analysis indicates that in general common classification algorithms perform better than clustering approaches.

Keywords

Text Segmentation, Multi-Author Decomposition, Parse Trees, pq-grams, Clustering

1. INTRODUCTION

The growing amount of currently available data is hardly manageable without the use of specific tools and algorithms that provide relevant portions of that data to the user. While this problem is generally addressed with information retrieval approaches, another possibility to significantly reduce the amount of data is to build clusters. Within each cluster, the data is similar according to some predefined features. Thereby many approaches exist that propose algorithms to cluster plain text documents (e.g. [16], [22]) or specific web documents (e.g. [33]) by utilizing various features.

Approaches which attempt to divide a single text document into distinguishable units like different topics, for example, are usually referred to as *text segmentation* approaches. Here, also many features including statistical models, similarities between words or other semantic analyses are used. Moreover, text clusters are also used in recent plagiarism detection algorithms (e.g. [34]) which try to build a cluster for the main author and one or more clusters for intrusive paragraphs. Another scenario where the clustering of text is applicable is the analysis of multi-author academic papers: especially the verification of collaborated student works such as bachelor or master theses can be useful in order to determine the amount of work done by each student.

Using results of previous work in the field of intrinsic plagiarism detection [31] and authorship attribution [32], the assumption that individual authors have significantly different writing styles in terms of the syntax that is used to construct sentences has been reused. For example, the following sentence (extracted from a web blog): *"My chair started squeaking a few days ago and it's driving me nuts."* (S1) could also be formulated as *"Since a few days my chair is squeaking - it's simply annoying."* (S2) which is semantically equivalent but differs significantly according to the syntax as can be seen in Figure 1. The main idea of this work is to quantify those differences by calculating grammar profiles and to use this information to decompose a collaboratively written document, i.e., to assign each paragraph of a document to an author.

The rest of this paper is organized as follows: Section 2 at first recapitulates the principle of pq-grams, which represent a core concept of the approach. Subsequently the algorithm is presented in detail, which is then evaluated in Section 3 by using different clustering algorithms and data sets. A comparison of clustering and classification approaches is discussed in Section 4, while Section 5 depicts related work. Finally, a conclusion and future work directions are given in Section 6.

2. ALGORITHM

In the following the concept of pq-grams is explained, which serves as the basic stylistic measure in this approach to distinguish between authors. Subsequently, the concrete steps performed by the algorithm are discussed in detail.

2.1 Preliminaries: pq-grams

Similar to n-grams that represent subparts of given length n of a string, pq-grams extract substructures of an ordered, labeled tree [4]. The size of a pq-gram is determined by a stem (p) and a base (q) like it is shown in Figure 2. Thereby p defines how much nodes are included vertically, and q defines the number of nodes to be considered horizontally. For example, a valid pq-gram with $p = 2$ and $q = 3$ starting from `PP` at the left side of tree (S2) shown in Figure 1 would be `[PP-NP-DT-JJ-NNS]` (the concrete words are omitted).

The pq-gram index then consists of all possible pq-grams of a tree. In order to obtain all pq-grams, the base is shifted left and right additionally: If then less than p nodes exist horizontally, the corresponding place in the pq-gram is filled with $\star$, in-

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

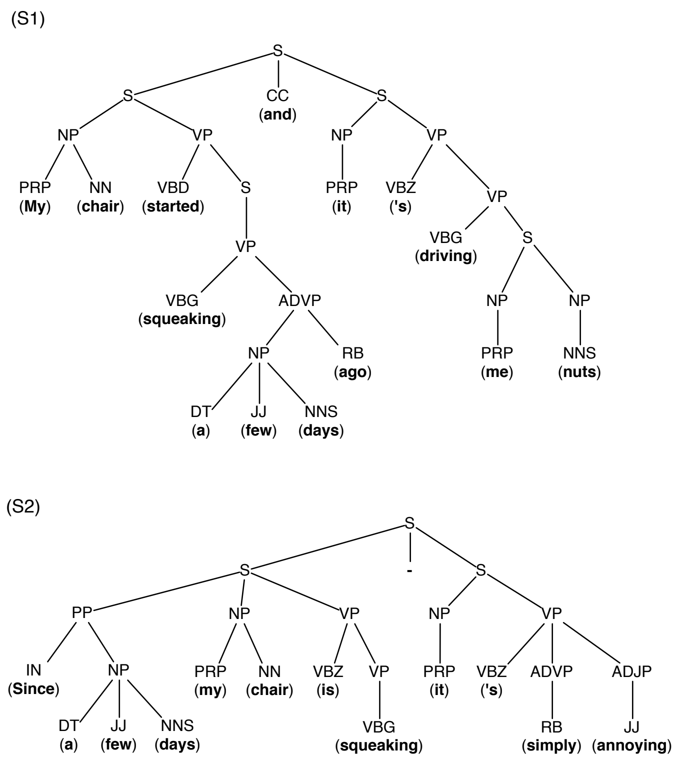


Figure 1: Grammar Trees of the Semantically Equivalent Sentences (S1) and (S2).

dicating a missing node. Applying this idea to the previous example, also the pq-gram `[PP-IN-⋆-⋆-⋆]` (no nodes in the base) is valid, as well as `[PP-NP-⋆-⋆-DT]` (base shifted left by two), `[PP-NP-⋆-DT-JJ]` (base shifted left by one), `[PP-NP-JJ-NNS-⋆]` (base shifted right by one) and `[PP-NP-NNS-⋆-⋆]` (base shifted right by two) have to be considered. As a last example, all leaves have the pq-gram pattern `[`*leaf_label*`-⋆-⋆-⋆-⋆]`.

Finally, the pq-gram index is the set of all valid pq-grams of a tree, whereby multiple occurrences of the same pq-grams are also present multiple times in the index.

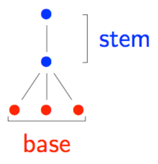


Figure 2: Structure of a pq-gram Consisting of Stem $p = 2$ and Base $q = 3$.

2.2 Clustering by Authors

The number of choices an author has to formulate a sentence in terms of grammar structure is rather high, and the assumption in this approach is that the concrete choice is made mostly intuitively and unconsciously. On that basis the grammar of authors is analyzed, which serves as input for common state-of-the-art clustering algorithms to build clusters of text documents or paragraphs. The decision of the clustering algorithms is thereby based on the frequencies of occurring pq-grams, i.e., on pq-gram profiles. In detail, given a text document the algorithm consists of the following steps:

1. At first the document is preprocessed by eliminating unnecessary whitespaces or non-parsable characters. For example, many data sets often are based on novels and articles of various authors, whereby frequently OCR text recognition is used due to the lack of digital data. Additionally, such documents contain problem sources like chapter numbers and titles or incorrectly parsed picture frames that result in non-alphanumeric characters.

2. Subsequently, the document is partitioned into single paragraphs. For simplification reasons this is currently done by only detecting multiple line breaks.

3. Each paragraph is then split into single sentences by utilizing a sentence boundary detection algorithm implemented within the *OpenNLP* framework[1]. Then for each sentence a full grammar tree is calculated using the *Stanford Parser* [19]. For example, Figure 1 depicts the grammar trees resulting from analyzing sentences (S1) and (S2), respectively. The labels of each tree correspond to a part-of-speech (POS) tag of the Penn Treebank set [23], where e.g *NP* corresponds to a noun phrase, *DT* to a determiner or *JJS* to a superlative adjective. In order to examine the building structure of sentences only like it is intended by this work, the concrete words, i.e., the leafs of the tree, are omitted.

4. Using the grammar trees of all sentences of the document, the pq-gram index is calculated. As shown in Section 2.1 all valid pq-grams of a sentence are extracted and stored into a pq-gram index. By combining all pq-gram indices of all sentences, a pq-gram profile is computed which contains a list of all pq-grams and their corresponding frequency of appearance in the text. Thereby the frequency is normalized by the total number of all appearing pq-grams. As an example, the five mostly used pq-grams using $p = 2$ and $q = 3$ of a sample document are shown in Table 1. The profile is sorted descending by the normalized occurrence, and an additional rank value is introduced that simply defines a natural order which is used in the evaluation (see Section 3).

Table 1: Example of the Five Mostly Used pq-grams of a Sample Document.

pq-gram	Occurrence [%]	Rank
NP-NN-⋆-⋆-⋆	2.68	1
PP-IN-⋆-⋆-⋆	2.25	2
NP-DT-⋆-⋆-⋆	1.99	3
NP-NNP-⋆-⋆-⋆	1.44	4
S-VP-⋆-⋆-VBD	1.08	5

5. Finally, each paragraph-profile is provided as input for clustering algorithms, which are asked to build clusters based on the pq-grams contained. Concretely, three different feature sets have been evaluated: (1.) the frequencies of occurences of each pq-gram, (2.) the rank of each pq-gram and (3.) a union of the latter sets.

[1]Apache OpenNLP, http://incubator.apache.org/opennlp, visited July 2014

2.3 Utilized Algorithms

Using the WEKA framework [15], the following clustering algorithms have been evaluated: K-Means [3], Cascaded K-Means (the number of clusters is cascaded and automatically chosen) [5], X-Means [26], Agglomerative Hierarchical Clustering [25], and Farthest First [9].

For the clustering algorithms *K-Means*, *Hierarchical Clustering* and *Farthest First* the number of clusters has been predefined according to the respective test data. This means if the test document has been collaborated by three authors, the number of clusters has also been set to three. On the other hand, the algorithms *Cascaded K-Means* and *X-Means* implicitly decide which amount of clusters is optimal. Therefore these algorithms have been limited only in ranges, i.e., the minimum and maximum number of clusters has been set to two and six, respectively.

3. EVALUATION

The utilization of pq-gram profiles as input features for modern clustering algorithms has been extensively evaluated using different documents and data sets. As clustering and classification problems are closely related, the global aim was to experiment on the accuracy of automatic text clustering using solely the proposed grammar feature, and furthermore to compare it to those of current classification techniques.

3.1 Test Data and Experimental Setup

In order to evaluate the idea, different documents and test data sets have been used, which are explained in more detail in the following. Thereby single documents have been created which contain paragraphs written by different authors, as well as multiple documents, whereby each document is written by one author. In the latter case, every document is treated as one (large) paragraph for simplification reasons.

For the experiment, different parameter settings have been evaluated, i.e., the pq-gram values p and q have been varied from 2 to 4, in combination with the three different feature sets. Concretely, the following data sets have been used:

- **Twain-Wells (T-W)**: This document has been specifically created for the evaluation of in-document clustering. It contains 50 paragraphs of the book *"The Adventures of Huckleberry Finn"* by Mark Twain, and 50 paragraphs of *"The Time Machine"* by H. G. Wells[2]. All paragraphs have been randomly shuffled, whereby the size of each paragraph varies from approximately 25 words up to 280 words.

- **Twain-Wells-Shelley (T-W-S)**: In a similar fashion a three-author document has been created. It again uses (different) paragraphs of the same books by Twain and Wells, and appends it by paragraphs of the book *"Frankenstein; Or, The Modern Prometheus"* by Mary Wollstonecraft Shelley. Summarizing, the document contains 50 paragraphs by Mark Twain, 50 paragraphs by H. G. Wells and another 50 paragraphs by Mary Shelley, whereby the paragraph sizes are similar to the Twain-Wells document.

- **The Federalist Papers (FED)**: Probably the mostly referred text corpus in the field of authorship attribution is a series of 85 political essays called "The Federalist Papers" written by John Jay, Alexander Hamilton and James Madison in the 18th century. While most of the authorships are undoubted, many works have studied and questioned the correct authorship of 12 disputed essays [24], which have been excluded in the experiment.

- **The PAN'12 competition corpus (PAN12)**: As a well-known, state-of-the-art corpus originally created for the use in authorship identification, parts[3] of the PAN2012 corpus [18] have been integrated. The corpus is composed of several fiction texts and split into several subtasks that cover small- and common-length documents (1800-6060 words) as well as larger documents (up to 13000 words) and novel-length documents (up to 170,000 words). Finally, the test setused in this evaluation contains 14 documents (paragraphs) written by three authors that are distributed equally.

3.2 Results

The best results of the evaluation are presented in Table 2, where the best performance for each clusterer over all data sets is shown in subtable (a), and the best configuration for each data set is shown in subtable (b), respectively. With an accuracy of 63.7% the K-Means algorithm worked best by using $p = 2, q = 3$ and by utilizing all available features. Interestingly, the X-Means algorithm also achieved good results considering the fact that in this case the number of clusters has been assigned automatically by the algorithm. Finally, the hierarchical cluster performed worst gaining an accuracy of nearly 10% less than K-Means.

Regarding the best performances for each test data set, the results for the manually created data sets from novel literature are generally poor. For example, the best result for the two-author document Twain-Wells is only 59.6%, i.e., the accuracy is only slightly better than the baseline percentage of 50%, which can be achieved by randomly assigning paragraphs into two clusters.[4] On the other hand, the data sets reused from authorship attribution, namely the FED and the PAN12 data set, achieved very good results with an accuracy of about 89% and 83%, respectively. Nevertheless, as the other data sets have been specifically created for the clustering evaluation, these results may be more expressive. Therefore a comparison between clustering and classification approaches is discussed in the following, showing that the latter achieve significantly better results on those data sets when using the same features.

Method	p	q	Feature Set	Accuracy
K-Means	3	2	All	**63.7**
X-Means	2	4	Rank	**61.7**
Farthest First	4	2	Occurrence-Rate	**58.7**
Cascaded K-Means	2	2	Rank	**55.3**
Hierarchical Clust.	4	3	Occurrence-Rate	**54.7**

(a) Clustering Algorithms

Data Set	Method	p	q	Feat. Set	Accuracy
T-W	X-Means	3	2	All	**59.6**
T-W-S	X-Means	3	4	All	**49.0**
FED	Farth. First	4	3	Rank	**89.4**
PAN12-A/B	K-Means	3	3	All	**83.3**

(b) Test Data Sets

Table 2: Best Evaluation Results for Each Clustering Algorithm and Test Data Set in Percent.

[2]The books have been obtained from the Project Gutenberg library, `http://www.gutenberg.org`, visited July 2014

[3]the subtasks A and B, respectively

[4]In this case X-Means dynamically created two clusters, but the result is still better than that of other algorithms using a fixed number of clusters.

4. COMPARISON OF CLUSTERING AND CLASSIFICATION APPROACHES

For the given data sets, any clustering problem can be rewritten as classification problem with the exception that the latter need training data. Although a direct comparison should be treated with caution, it still gives an insight of how the two different approaches perform using the same data sets. Therefore an additional evaluation is shown in the following, which compares the performance of the clustering algorithms to the performance of the the following classification algorithms: Naive Bayes classifier [17], Bayes Network using the K2 classifier [8], Large Linear Classification using LibLinear [12], Support vector machine using LIBSVM with nu-SVC classification [6], k-nearest-neighbors classifier (kNN) using $k = 1$ [1], and a pruned C4.5 decision tree (J48) [28]. To compensate the missing training data, a 10-fold cross-validation has been used for each classifier.

Table 3 shows the performance of each classifier compared to the best clustering result using the same data and pq-setting. It can be seen that the classifiers significantly outperform the clustering results for the Twain-Wells and Twain-Wells-Shelley documents. The support vector machine framework (LibSVM) and the linear classifier (LibLinear) performed best, reaching a maximum accuracy of nearly 87% for the Twain-Wells document. Moreover, the average improvement is given in the bottom line, showing that most of the classifiers outperform the best clustering result by over 20% in average. Solely the kNN algorithm achieves minor improvements as it attributed the two-author document with a poor accuracy of about 60% only.

A similar general improvement could be achieved on the three-author document Twain-Wells-Shelley as can be seen in subtable (b). Again, LibSVM could achieve an accuracy of about 75%, whereas the best clustering configuration could only reach 49%. Except for the kNN algorithm, all classifiers significantly outperform the best clustering results for every configuration.

Quite different comparison results have been obtained for the Federalist Papers and PAN12 data sets, respectively. Here, the improvements gained from the classifiers are only minor, and in some cases are even negative, i.e., the classification algorithms perform worse than the clustering algorithms. A general explanation is the good performance of the clustering algorithms on these data sets, especially by utilizing the Farthest First and K-Means algorithms.

In case of the Federalist Papers data set shown in subtable (c), all algorithms except kNN could achieve at least some improvement. Although the LibLinear classifier could reach an outstanding accuracy of 97%, the global improvement is below 10% for all classifiers. Finally, subtable (d) shows the results for PAN12, where the outcome is quite diverse as some classifiers could improve the clusterers significantly, whereas others worsen the accuracy even more drastically. A possible explanation might be the small data set (only the subproblems A and B have been used), which may not be suited very well for a reliable evaluation of the clustering approaches.

Summarizing, the comparison of the different algorithms reveal that in general classification algorithms perform better than clustering algorithms when provided with the same (pq-gram) feature set. Nevertheless, the results of the PAN12 experiment are very diverse and indicate that there might be a problem with the data set itself, and that this comparison should be treated carefully.

p	q	Algorithm	Max	N-Bay	Bay-Net	LibLin	LibSVM	kNN	J48
2	2	X-Means	57.6	77.8	82.3	85.2	86.9	62.6	85.5
2	3	X-Means	56.6	79.8	80.8	81.8	83.3	60.6	80.8
2	4	X-Means	57.6	76.8	79.8	82.2	83.8	58.6	81.0
3	2	X-Means	59.6	78.8	80.8	81.8	83.6	59.6	80.8
3	3	X-Means	53.5	76.8	77.8	80.5	82.3	61.6	79.8
3	4	X-Means	52.5	81.8	79.8	81.8	83.8	63.6	82.0
4	2	K-Means	52.5	86.9	83.3	83.5	84.3	62.6	81.8
4	3	X-Means	52.5	79.8	79.8	80.1	80.3	59.6	77.4
4	4	Farth. First	51.5	72.7	74.7	75.8	77.0	60.6	75.8
		average improvement		**24.1**	**25.0**	**26.5**	**27.9**	**6.2**	**25.7**

(a) Twain-Wells

p	q	Algorithm	Max	N-Bay	Bay-Net	LibLin	LibSVM	kNN	J48
2	2	K-Means	44.3	67.8	70.8	74.0	75.2	51.0	73.3
2	3	X-Means	38.3	65.1	67.1	70.7	72.3	48.3	70.2
2	4	X-Means	45.6	63.1	68.1	70.5	71.8	49.0	69.3
3	2	X-Means	45.0	51.7	64.1	67.3	68.8	45.6	65.4
3	3	X-Means	47.0	57.7	64.8	67.3	68.5	47.0	65.9
3	4	X-Means	49.0	67.8	67.8	70.5	72.5	46.3	68.3
4	2	X-Means	36.2	61.1	67.1	69.1	69.5	50.3	65.1
4	3	K-Means	35.6	53.0	63.8	67.6	70.0	47.0	66.6
4	4	X-Means	35.6	57.7	66.1	68.5	69.3	42.3	66.8
		average improvement		**18.7**	**24.8**	**27.7**	**29.0**	**5.6**	**26.0**

(b) Twain-Wells-Shelley

p	q	Algorithm	Max	N-Bay	Bay-Net	LibLin	LibSVM	kNN	J48
2	2	Farth. First	77.3	81.1	86.4	90.9	84.2	74.2	81.8
2	3	Farth. First	78.8	85.6	87.4	92.4	89.0	78.8	82.8
2	4	X-Means	78.8	89.4	92.4	90.9	87.3	89.4	85.9
3	2	K-Means	81.8	82.6	87.9	92.4	85.5	80.3	83.8
3	3	K-Means	78.8	92.4	92.4	92.4	86.4	81.8	83.8
3	4	Farth. First	86.4	84.8	90.9	97.0	85.8	81.8	85.6
4	2	Farth. First	86.6	81.8	89.4	87.9	83.3	77.3	84.1
4	3	Farth. First	89.4	85.6	92.4	89.4	85.8	80.3	83.3
4	4	Farth. First	84.8	86.4	90.9	89.4	85.8	84.8	83.6
		average improvement		**3.0**	**7.5**	**8.9**	**3.4**	**-1.6**	**1.3**

(c) Federalist Papers

p	q	Algorithm	Max	N-Bay	Bay-Net	LibLin	LibSVM	kNN	J48
2	2	K-Means	83.3	83.3	33.3	100.0	100.0	100.0	33.3
2	3	K-Means	83.3	83.3	33.3	100.0	100.0	100.0	33.3
2	4	K-Means	83.3	83.4	33.3	100.0	100.0	100.0	33.3
3	2	K-Means	83.3	75.0	33.3	91.7	91.7	100.0	33.3
3	3	K-Means	83.3	100.0	33.3	100.0	91.7	100.0	33.3
3	4	Farth. First	75.0	66.7	33.3	100.0	100.0	91.7	33.3
4	2	K-Means	83.3	91.7	33.3	91.7	75.0	91.7	33.3
4	3	K-Means	83.3	75.0	33.3	100.0	75.0	91.7	33.3
4	4	K-Means	83.3	75.0	33.3	100.0	83.4	83.4	33.3
		average improvement		**-0.9**	**-49.1**	**15.8**	**8.4**	**13.0**	**-49.1**

(d) PAN12-A/B

Table 3: Best Evaluation Results for each Clustering Algorithm and Test Data Set in Percent.

5. RELATED WORK

Most of the traditional document clustering approaches are based on occurrences of words, i.e., inverted indices are built and used to group documents. Thereby a unit to be clustered conforms exactly to one document. The main idea is often to compute topically related document clusters and to assist web search engines to be able to provide better results to the user, whereby the algorithms proposed frequently are also patented (e.g. [2]). Regularly applied concepts in the feature extraction phase are the term frequency tf, which measures how often a word in a document occurs, and the term frequency-inverse document frequency $tf - idf$, which measures the significance of a word compared to the whole document collection. An example of a classical approach using these techniques is published in [21].

The literature on cluster analysis within a single document to discriminate the authorships in a multi-author document like it is done in this paper is surprisingly sparse. On the other hand, many approaches exist to separate a document into paragraphs of different topics, which are generally called *text segmentation* problems. In this domain, the algorithms often perform vocabulary analysis in various forms like word stem repetitions [27] or word frequency models [29], whereby *"methods for finding the topic boundaries include sliding window, lexical chains, dynamic programming, agglomerative clustering and divisive clustering"* [7]. Despite the given possibility to modify these techniques to also cluster by authors instead of topics, this is rarely done. In the following some of the existing methods are shortly summarized.

Probably one of the first approaches that uses stylometry to automatically detect boundaries of authors of collaboratively written text is proposed in [13]. Thereby the main intention was not to expose authors or to gain insight into the work distribution, but to provide a methodology for collaborative authors to equalize their style in order to achieve better readability. To extract the style of separated paragraphs, common stylometric features such as word/sentence lengths, POS tag distributions or frequencies of POS classes at sentence-initial and sentence-final positions are considered. An extensive experiment revealed that styolmetric features can be used to find authorship boundaries, but that there has to be done additional research in order to increase the accuracy and informativeness.

In [14] the authors also tried to divide a collaborative text into different single-author paragraphs. In contrast to the previously described handmade corpus, a large data set has been computationally created by using (well-written) articles of an internet forum. At first, different neural networks have been utilized using several stylometric features. By using 90% of the data for training, the best network could achieve an F-score of 53% for multi-author documents on the remaining 10% of test data. In a second experiment, only letter-bigram frequencies are used as distinguishing features. Thereby an authorship boundary between paragraphs was marked if the cosine distance exceeded a certain threshold. This method reached an F-score of only 42%, and it is suspected that letter-bigrams are not suitable for the (short) paragraphs used in the evaluation.

A two-stage process to cluster Hebrew Bible texts by authorship is proposed in [20]. Because a first attempt to represent chapters only by bag-of-words led to negative results, the authors additionally incorporated sets of synonyms (which could be generated by comparing the original Hebrew texts with an English translation). With a modified cosine-measure comparing these sets for given chapters, two core clusters are compiled by using the *ncut* algorithm [10]. In the second step, the resulting clusters are used as training data for a support vector machine, which finally assigns every chapter to one of the two core clusters by using the simple bag-of-words features tested earlier. Thereby it can be the case, that units originally assigned to one cluster are moved to the other one, depending on the prediction of the support vector machine. With this two-stage approach the authors report a good accuracy of about 80%, whereby it should be considered that the size of potential authors has been fixed to two in the experiment. Nevertheless, the authors state that their approach could be extended for more authors with less effort.

6. CONCLUSION AND FUTURE WORK

In this paper, the automatic creation of paragraph clusters based on the grammar of authors has been evaluated. Different state-of-the-art clustering algorithms have been utilized with different input features and tested on different data sets. The best working algorithm K-Means could achieve an accuracy of about 63% over all test sets, whereby good individual results of up to 89% could be reached for some configurations. On the contrary, the specifically created documents incorporating two and three authors could only be clustered with a maximum accuracy of 59%.

A comparison between clustering and classification algorithms using the same input features has been implemented. Disregarding the missing training data, it could be observed that classifiers generally produce higher accuracies with improvements of up to 29%. On the other hand, some classifiers perform worse on average than clustering algorithms over individual data sets when using some pq-gram configurations. Nevertheless, if the maximum accuracy for each algorithm is considered, all classifiers perform significantly better as can be seen in Figure 3. Here the best performances of all utilized classification and clustering algorithms are illustrated. The

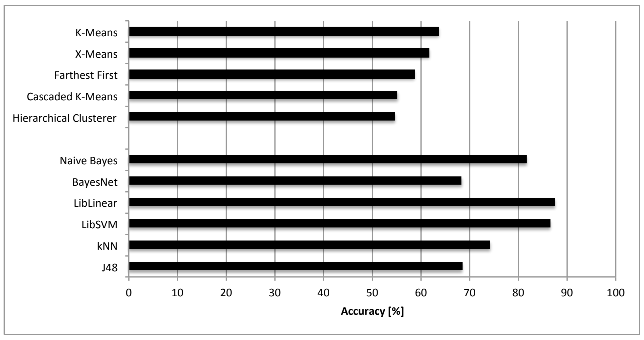


Figure 3: Best Evaluation Results Over All Data Sets For All Utilized Clustering and Classification Algorithms.

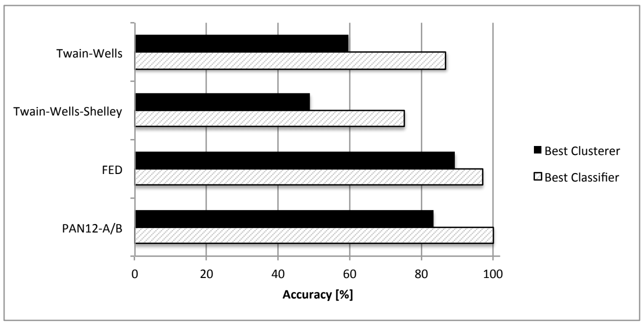


Figure 4: Best Clustering and Classification Results For Each Data Set.

linear classification algorithm LibLinear could reach nearly 88%, outperforming K-Means by 25% over all data sets.

Finally, the best classification and clustering results for each data set are shown in Figure 4. Consequently the classifiers achieve higher accuracies, whereby the PAN12 subsets could be classified 100% correctly. As can be seen, a major improvement can be gained for the novel literature documents. For example, the best classifier reached 87% on the Twain-Wells document, whereas the best clustering approach achieved only 59%.

As shown in this paper, paragraphs of documents can be split and clustered based on grammar features, but the accuracy is below that of classification algorithms. Although the two algorithm types should not be compared directly as they are designed to manage different problems, the significant differences in accuracies indicate that classifiers can handle the grammar features better. Nevertheless future work should focus on evaluating the same features on larger data sets, as clustering algorithms may produce better results with increasing amount of sample data.

Another possible application could be the creation of whole document clusters, where documents with similar grammar are grouped together. Despite the fact that such huge clusters are very difficult to evaluate - due to the lack of ground truth data - a navigation through thousands of documents based on grammar may be interesting like it has been done for music genres (e.g. [30]) or images (e.g. [11]). Moreover, grammar clusters may also be utilized for modern recommendation algorithms once they have been calculated for large data sets. For example, by analyzing all freely available books from libraries like Project Gutenberg, a system could recommend other books with a similar style based on the users reading history. Also, an enhancement of current commercial recommender systems that are used in large online stores like Amazon is conceivable.

7. REFERENCES

[1] D. Aha and D. Kibler. Instance-Based Learning Algorithms. *Machine Learning*, 6:37–66, 1991.

[2] C. Apte, S. M. Weiss, and B. F. White. Lightweight Document Clustering, Nov. 25 2003. US Patent 6,654,739.

[3] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[4] N. Augsten, M. Böhlen, and J. Gamper. The pq-Gram Distance between Ordered Labeled Trees. *ACM Transactions on Database Systems (TODS)*, 2010.

[5] T. Caliński and J. Harabasz. A Dendrite Method for Cluster Analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27, 1974.

[6] C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[7] F. Y. Choi. Advances in Domain Independent Linear Text Segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33. Association for Computational Linguistics, 2000.

[8] G. F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks From Data. *Machine learning*, 9(4):309–347, 1992.

[9] S. Dasgupta. Performance Guarantees for Hierarchical Clustering. In *Computational Learning Theory*, pages 351–363. Springer, 2002.

[10] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: Spectral Clustering and Normalized Cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556. ACM, 2004.

[11] A. Faktor and M. Irani. "Clustering by Composition" - Unsupervised Discovery of Image Categories. In *Computer Vision–ECCV 2012*, pages 474–487. Springer, 2012.

[12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[13] A. Glover and G. Hirst. Detecting Stylistic Inconsistencies in Collaborative Writing. In *The New Writing Environment*, pages 147–168. Springer, 1996.

[14] N. Graham, G. Hirst, and B. Marthi. Segmenting Documents by Stylistic Character. *Natural Language Engineering*, 11(04):397–415, 2005.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: an Update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[16] A. Hotho, S. Staab, and G. Stumme. Ontologies Improve Text Document Clustering. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 541–544. IEEE, 2003.

[17] G. H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[18] P. Juola. An Overview of the Traditional Authorship Attribution Subtask. In *CLEF (Online Working Notes/Labs/Workshop)*, 2012.

[19] D. Klein and C. D. Manning. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003.

[20] M. Koppel, N. Akiva, I. Dershowitz, and N. Dershowitz. Unsupervised Decomposition of a Document into Authorial Components. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1356–1364, Stroudsburg, PA, USA, 2011.

[21] B. Larsen and C. Aone. Fast and Effective Text Mining Using Linear-Time Document Clustering. In *Proceedings of the 5th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22. ACM, 1999.

[22] Y. Li, S. M. Chung, and J. D. Holt. Text Document Clustering Based on Frequent Word Meaning Sequences. *Data & Knowledge Engineering*, 64(1):381–404, 2008.

[23] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330, June 1993.

[24] F. Mosteller and D. Wallace. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley, 1964.

[25] F. Murtagh. A Survey of Recent Advances in Hierarchical Clustering Algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[26] D. Pelleg, A. W. Moore, et al. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *ICML*, pages 727–734, 2000.

[27] J. M. Ponte and W. B. Croft. Text Segmentation by Topic. In *Research and Advanced Technology for Digital Libraries*, pages 113–125. Springer, 1997.

[28] J. R. Quinlan. *C4.5: Programs for Machine Learning*, volume 1. Morgan Kaufmann, 1993.

[29] J. C. Reynar. Statistical Models for Topic Segmentation. In *Proc. of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 357–364, 1999.

[30] N. Scaringella, G. Zoia, and D. Mlynek. Automatic Genre Classification of Music Content: a Survey. *Signal Processing Magazine, IEEE*, 23(2):133–141, 2006.

[31] M. Tschuggnall and G. Specht. Using Grammar-Profiles to Intrinsically Expose Plagiarism in Text Documents. In *Proc. of the 18th Conf. of Natural Language Processing and Information Systems (NLDB)*, pages 297–302, 2013.

[32] M. Tschuggnall and G. Specht. Enhancing Authorship Attribution By Utilizing Syntax Tree Profiles. In *Proc. of the 14th Conf. of the European Chapter of the Assoc. for Computational Ling. (EACL)*, pages 195–199, 2014.

[33] O. Zamir and O. Etzioni. Web Document Clustering: A Feasibility Demonstration. In *Proc. of the 21st annual international ACM conference on Research and development in information retrieval (SIGIR)*, pages 46–54. ACM, 1998.

[34] D. Zou, W.-J. Long, and Z. Ling. A Cluster-Based Plagiarism Detection Method. In *Notebook Papers of CLEF 2010 LABs and Workshops, 22-23 September*, 2010.

Proaktive modellbasierte Performance-Analyse und -Vorhersage von Datenbankanwendungen

Christoph Koch

Friedrich-Schiller-Universität Jena
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 2
07743 Jena

Christoph.Koch@uni-jena.de

DATEV eG
Abteilung Datenbanken
Paumgartnerstr. 6 - 14
90429 Nürnberg

Christoph.Koch@datev.de

KURZFASSUNG

Moderne (Datenbank-)Anwendungen sehen sich in der heutigen Zeit mit immer höheren Anforderungen hinsichtlich Flexibilität, Funktionalität oder Verfügbarkeit konfrontiert. Nicht zuletzt für deren Backend – ein meist relationales Datenbankmanagementsystem – entsteht dadurch eine kontinuierlich steigende Komplexität und Workload, die es frühestmöglich proaktiv zu erkennen, einzuschätzen und effizient zu bewältigen gilt. Die dazu nötigen Anwendungs- und Datenbankspezialisten sind jedoch aufgrund immer engerer Projektpläne, kürzerer Release-Zyklen und weiter wachsender Systemlandschaften stark ausgelastet, sodass für regelmäßige proaktive Expertenanalysen hinsichtlich der Datenbank-Performance kaum Kapazität vorhanden ist.

Zur Auflösung dieses Dilemmas stellt dieser Beitrag ein Verfahren vor, mit dessen Hilfe frühzeitig auf Grundlage der Datenmodellierung und synthetischer Datenbankstatistiken Performance-Analysen und -Vorhersagen für Anwendungen mit relationalem Datenbank-Backend durchgeführt und deren Ergebnisse auf leicht zugängliche Weise visualisiert werden können.

Kategorien und Themenbeschreibung

Data Models and Database Design, Database Performance

Allgemeine Bestimmungen

Performance, Design

Schlüsselwörter

Performance, Proaktivität, Statistiken, relationale Datenbanken, Modellierung, UML, Anwendungsentwicklung

1. EINLEITUNG

Zur Erfüllung komplexerer Anforderungen und maximalen Benutzerkomforts ist gute Performance eine Grundvoraussetzung für moderne Datenbankanwendungen. Neben Anwendungs-Design und Infrastrukturkomponenten wie Netzwerk oder Anwendungs- beziehungsweise Web-Server wird sie maßgeblich durch die Performance ihres Datenbank-Backends – wir beschränken uns hier ausschließlich auf relationale Datenbankmanagementsysteme (DBMS) – bestimmt [1]. Dabei ist die Datenbank-Performance einer Anwendung selbst ebenfalls durch zahlreiche Faktoren beeinflusst. Während Hardware- und systemseitige Eigenschaften oftmals durch bestehende Infrastrukturen vorgegeben sind, können speziell das Datenbank-Design sowie die anwendungsseitig implementierten Zugriffe mittels SQL weitgehend frei gestaltet werden. Hinzu kommt als Einflussfaktor noch die Beschaffenheit der zu speichernden/gespeicherten Daten, die sich in Menge und Verteilung ebenfalls stark auf die Performance auswirkt.

Das **Datenbank-Design** entwickelt sich über unterschiedlich abstrakte, aufeinander aufbauende Modellstrukturen vom konzeptionellen hin zum physischen Datenmodell. Bereits bei der Entwicklung dieser Modelle können „Designfehler“ wie beispielsweise fehlende oder „übertriebene“ Normalisierungen gravierende Auswirkungen auf die späteren Antwortzeiten des Datenbanksystems haben. Der Grad an Normalisierung selbst ist jedoch nur als vager Anhaltspunkt für die Performance von Datenbanksystemen anzusehen, der sich ab einem gewissen Maß auch negativ auswirken kann. Eine einfache Metrik zur Beurteilung der Qualität des Datenbank-Designs bezüglich der zu erwartenden Performance (in Abhängigkeit anderer Einflussfaktoren, wie etwa der Workload) existiert nach vorhandenem Kenntnisstand nicht.

Etwas abweichend dazu verhält es sich mit dem Einfluss der **Workload** – repräsentiert als Menge von SQL-Statements und der Häufigkeit ihrer Ausführung, die von der Anwendung an das Datenbanksystem zum Zugriff auf dort gespeicherte Daten abgesetzt wird. Moderne DBMS besitzen einen kostenbasierten Optimierer zur Optimierung eingehender Statements. Dieser berechnet mögliche Ausführungspläne und wählt unter Zuhilfenahme von gesammelten Objekt-Statistiken den günstigsten Ausführungsplan zur Abarbeitung eines SQL-Statements aus. Mittels DBMS-internen Mechanismen – im Folgenden als

EXPLAIN-Mechanismen bezeichnet – besteht die Möglichkeit, noch vor der eigentlichen Ausführung von Statements den vom Optimierer bestimmten optimalen Ausführungsplan ermitteln und ausgeben zu lassen. Zusätzlich umfasst das EXPLAIN-Ergebnis eine Abschätzung der zur Abarbeitung des Ausführungsplans erwarteten Zugriffskosten bezüglich der CPU-und I/O-Zeit – fortan als Kosten bezeichnet. Anhand dieser Informationen können bereits frühzeitig in Hinblick auf die Datenbank-Performance (häufige) teure Zugriffe erkannt und gegebenenfalls optimiert werden. Voraussetzung für dieses Vorgehen ist allerdings, dass dem DBMS zur Berechnung der Ausführungspläne repräsentative **Datenbank-Statistiken** vorliegen, was insbesondere für neue Datenbankanwendungen nicht der Fall ist.

Auf der anderen Seite sehen sich sowohl Anwendungsentwickler- beziehungsweise -designerteams als auch Datenbankspezialisten mit immer komplexeren Anforderungen und Aufgaben konfrontiert. Kapazitäten für umfangreiche Performance-Analysen oder auch nur die Aneignung des dafür nötigen Wissens sind oft nicht gegeben. Nicht zuletzt deshalb geraten proaktive Performance-Analysen verglichen mit beispielsweise funktionalen Tests vermehrt aus dem Fokus.

Das im vorliegenden Beitrag vorgestellte modellbasierte Konzept setzt an diesen beiden Problemen an und stellt Mechanismen vor, um auf einfache Art und Weise eine repräsentative proaktive Analyse der Datenbank-Performance zu ermöglichen. Nachdem in **Kapitel 2** eine Abgrenzung zu alternativen/verwandten Ansätzen gegeben wird, rückt **Kapitel 3** den Entwicklungsprozess einer Datenbank-Anwendung in den Fokus. **Kapitel 4** beschäftigt sich mit dem entwickelten proaktiven Ansatz und stellt wesentliche Schritte/Komponenten vor. Abschließend fasst **Kapitel 5** den Beitrag zusammen.

2. VERWANDTE ARBEITEN

Das Ziel des im Beitrag vorgestellten proaktiven Ansatzes zur Performance-Analyse und -Vorhersage von Datenbankanwendungen ist die **frühzeitige Erkennung von potentiellen Performance-Problemen** auf Basis einer möglichst effizienten, leicht verständlichen Methodik. Dies verfolgt auch der Ansatz von [2], dessen Grundprinzip – Informationen über Daten und Datenzugriffe, die aus der Anforderungsanalyse einer Anwendung bekannt sind, zur frühzeitigen Optimierung zu nutzen – sich auch im vorliegenden Beitrag wiederfindet. Dabei gelangt [2] durch eine eigene, dem Datenbank-Optimierer nachempfundene Logik und dem verwendeten Modell des offenen Warteschlangennetzwerks frühzeitig zu Kostenabschätzungen bezüglich der Datenbank-Performance. Das in Kapitel 4 des vorliegenden Beitrags vorgestellte Konzept nutzt dagegen synthetisch erzeugte Statistiken und datenbankinterne EXPLAIN-Mechanismen, um eine kostenmäßige Performance-Abschätzung zu erhalten. Damit berücksichtigt es stets sowohl aktuelle als auch zukünftige Spezifika einzelner Datenbank-Optimierer und bleibt entgegen [2] von deren interner Berechnungslogik unabhängig. Ein weiterer Unterschied zwischen beiden Ansätzen besteht in der Präsentation der Analyseergebnisse. Während sich [2] auf tabellarische Darstellungen beschränkt, nutzt das im Beitrag vorstellte Konzept eine auf der Grundlage der Unified Modeling Language (UML) visualisierte Darstellungsform.

Ähnlich wie [2] basieren auch weitere **Ansätze zur Performance-Analyse und -Evaluation** auf dem Modell des Warteschlangennetzwerks. Ein Überblick dazu findet sich in [3]. Demnach zeigt sich für all diese Konzepte ein eher wissenschaftlicher Fokus und eine damit einhergehende weitgehend unerprobte Übertragbarkeit auf die Praxis. So fehlen Studien zur Integration in praxisnahe (Entwicklungs-)Prozesse, zur Benutzerfreundlichkeit sowie zum Kosten-Nutzen-Verhältnis der notwendigen Maßnahmen. Ein damit einhergehendes Defizit ist zusätzlich der mangelnde Toolsupport. Das in Kapitel 4 vorgestellte Konzept verfolgt diesbezüglich einen davon abweichenden Ansatz. Es baut direkt auf etablierten modellbasierten Praxisabläufen bei der Entwicklung von Datenbankanwendungen auf (vgl. Kapitel 3). Insbesondere durch die Verwendung von standardisierten UML-Erweiterungsmechanismen integriert es sich auch Tool-seitig nahtlos in bestehende UML-unterstützende Infrastrukturen.

Die Methodik der **synthetischen Statistiken** – also dem künstlichen Erstellen sowie Manipulieren von Datenbank-Statistiken – ist neben dem in Kapitel 4 vorgestellten Ansatz wesentlicher Bestandteil von [4]. Sie wird zum einen verwendet, um Statistiken aus Produktionsumgebungen in eine Testumgebung zu transferieren. Zum anderen sieht der Ansatz aber auch die gezielte manuelle Veränderung der Statistiken vor, um mögliche dadurch entstehende Änderungen in den Ausführungsplänen und den zu deren Abarbeitung benötigten Kosten mithilfe anschließender EXPLAIN-Analysen feststellen zu können. Dies kann beispielsweise bezogen auf Statistiken zur Datenmenge dafür genutzt werden, um Zugriffe auf eine (noch) kleine Tabelle mit wenigen Datensätzen bereits so zu simulieren, als ob diese eine enorme Menge an Daten umfasst. Weitere Einbettungen in den Entwicklungsprozess von Datenbankanwendungen sieht [4] gegenüber dem hier vorgestellten Ansatz allerdings nicht vor.

Ein weiterer Ansatzpunkt zur Performance-Analyse und -Optimierung existiert im Konzept des **autonomen Datenbank-Tunings** [5][6],[7] – also dem fortlaufenden Optimieren des physischen Designs von bereits bestehenden Datenbanken durch das DBMS selbst. Ein autonomes System erkennt anhand von erlerntem Wissen potentielle Probleme und leitet passende Optimierungsmaßnahmen ein, bevor sich daraus negative Auswirkungen ergeben. Dazu zählt beispielsweise die autonome Durchführung einer Reorganisierung von Daten, um fortwährend steigenden Zugriffszeiten entgegenzuwirken. Ähnlich können auch die mittlerweile je System vielseitig vorhandenen Tuning-Advisor wie beispielsweise [8] und [9] angesehen werden, die zwar nicht automatisch optimierend ins System eingreifen, dem Administrator aber Empfehlungen zu sinnvoll durchzuführenden Aktionen geben. Sowohl das autonome Tuning als auch die Tuning-Advisor sind nicht als Alternative zu dem im vorliegenden Beitrag vorgestellten Ansatz einzuordnen. Vielmehr können sich diese Konzepte ergänzen, indem die Anwendungsentwicklung auf Basis des in Kapitel 4 vorgestellten Konzepts erfolgt und für die spätere Anwendungsadministration/ -evolution verschiedene Tuning-Advisor und die Mechanismen des autonomen Tunings zum Einsatz kommen.

3. ENTWICKLUNGSPROZESS VON DATENBANKANWENDUNGEN

Der Entwicklungsprozess von Anwendungen lässt sich anhand des **System Development Lifecycle (SDLC)** beschreiben und in verschiedene Phasen von der Analyse der Anforderungen bis hin zum Betrieb/zur Wartung der fertigen Software gliedern [1].

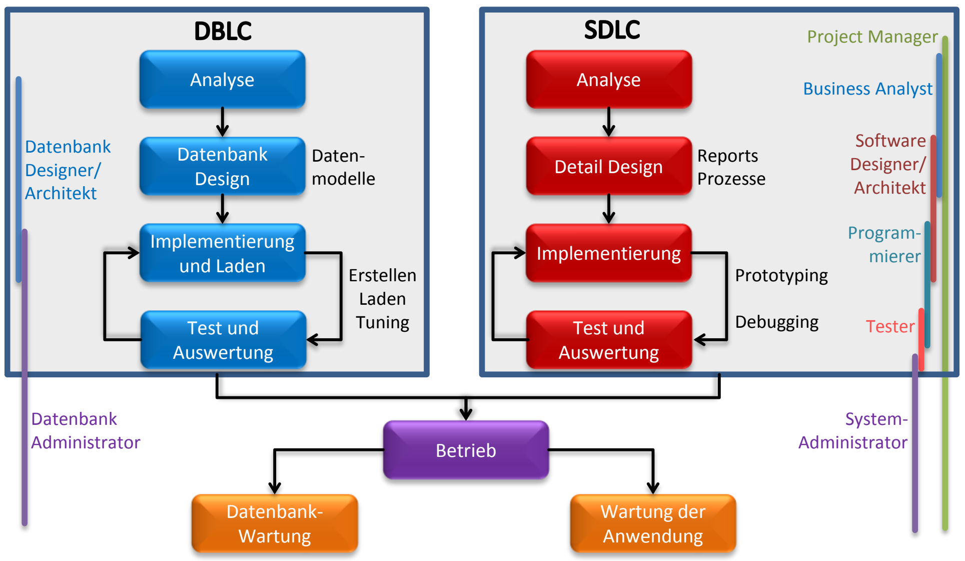


Abbildung 1: Phasen und Akteure im Database und Software Development Lifecycle (DBLC und SDLC)

Zusätzlich zur reinen Anwendungsentwicklung sind weitere Abläufe zur Planung und Bereitstellung einer geeigneten Infrastruktur nötig. Für Datenbankanwendungen wäre das unter anderem der Entwicklungsprozess der Datenbank, welcher sich nach [1] ebenfalls durch ein dem SDLC ähnliches Modell – dem Database Lifecycle (DBLC) – formalisieren lässt. Beide Entwicklungsprozesse verlaufen zeitlich parallel und werden insbesondere in größeren Unternehmen/Projekten durch verschiedene Akteure realisiert. Auf Grundlage von [1] liefert *Abbildung 1* eine Übersicht dazu. Sie visualisiert parallel ablaufende Entwicklungsphasen und eine Auswahl an zuständigen Akteuren, deren konkrete Zusammensetzung/Aufgabenverteilung aber stark abhängig von der Projektgröße und dem Projektteam ist. Wichtig sind hier besonders zwei Erkenntnisse. Zum einen finden ähnliche Entwicklungsprozesse bei Anwendung und Datenbank parallel statt – in etwa das Anwendungsdesign und das Datenbankdesign. Zum anderen können sehr viele Akteure am gesamten Entwicklungsprozess beteiligt sein, sodass Designer, Programmierer, Tester und Administratoren in der Regel disjunkte Personenkreise bilden.

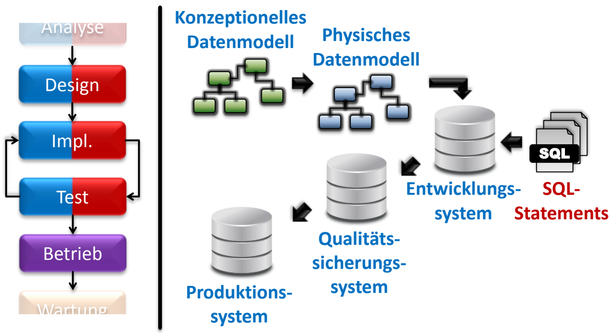


Abbildung 2: Performance-relevante Entwicklungsschritte

Aus dem Blickwinkel der Datenbank-Performance und der darauf einwirkenden bereits genannten Einflussfaktoren reduziert sich der Entwicklungsprozess von Datenbankanwendungen auf die in *Abbildung 2* visualisierten Aufgaben. Anhand der **analysierten Anforderungen** wird im **Datenbank-Design** ein konzeptionelles Datenmodell entwickelt, das anschließend hin zum physischen Datenmodell verfeinert wird. Da sich der Beitrag auf die in der Praxis vorherrschenden relationalen DBMS beschränkt, wird auf das in der Theorie gebräuchliche Zwischenprodukt des logischen Datenmodells (relationale Abbildung) verzichtet.

Nachdem die Design-Phase abgeschlossen ist, beginnt die **Implementierung**. Datenbankseitig wird dabei das physische Datenmodell mittels Data Definition Language (DDL) in ein Datenbankschema innerhalb eines installierten und geeignet konfigurierten DBMS umgesetzt und möglicherweise vorhandene Testdaten geladen. Anwendungsseitig erfolgt parallel dazu die Entwicklung von SQL-Statements zum Zugriff auf die Datenbank sowie die Implementierung der Anwendung selbst. Nach Fertigstellung einzelner Module finden mithilfe des Entwicklungs- und Qualitätssicherungssystems **kontinuierliche Tests** statt, die sich allerdings anfangs auf die Prüfung funktionaler Korrektheit beschränken. Performance-Untersuchungen, insbesondere bezogen auf die Datenbankzugriffe, erfolgen in der Regel erst gezielt zum Abschluss der Implementierungphase mittels aufwändig vorzubereitender und im Qualitätssicherungssystem durchzuführender Lasttests.

Die Folgen aus diesem Vorgehen für die Erkennung und Behandlung von Performance-Problemen sind mitunter gravierend. Engpässe werden erst spät (im **Betrieb**) bemerkt und sind aufgrund des fortgeschrittenen Entwicklungsprozesses nur mit hohem Aufwand zu korrigieren. Basieren sie gar auf unvorteilhaften Design-Entscheidungen beispielsweise bezogen auf die Datenmodellierung, ist eine nachträgliche Korrektur aufgrund zahlreicher Abhängigkeiten (Anwendungslogik, SQL-Statements, Testdatenbestände, etc.), getrennten Zuständigkeiten und in der Regel engen Projektzeitplänen nahezu ausgeschlossen. Erfahrungen aus dem Arbeitsumfeld des Autors haben dies wiederholt bestätigt.

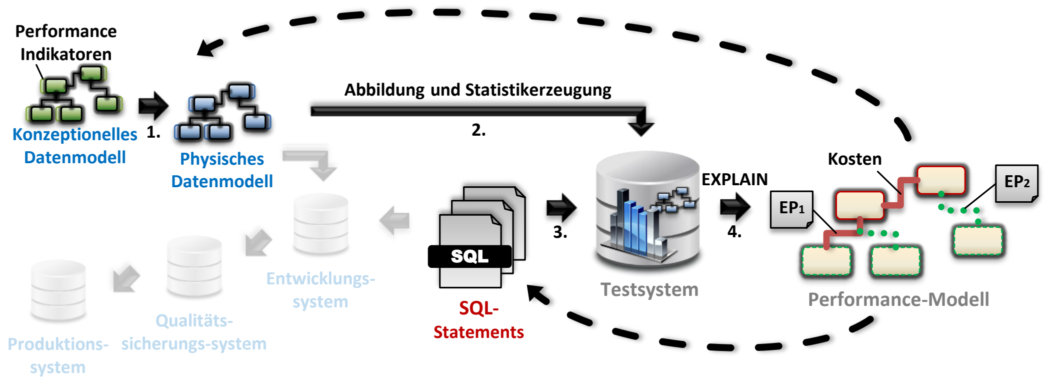


Abbildung 3: Ansatz zur proaktiven modellbasierten Performance-Analyse und -Vorhersage

4. PROAKTIVE MODELLBASIERTE PERFORMANCE-ANALYSE

Alternativ zur Performance-Analyse mittels Lasttests (vgl. Kapitel 3) bieten sich zur Kontrolle der SQL-Performance die eingangs erwähnten **EXPLAIN-Mechanismen** an. Mit deren Hilfe lassen sich bei vorliegendem physischen Datenbank-Design (inklusive Indexe, etc.) bereits in frühen Abschnitten der Implementierungsphase Auswertungen zu Ausführungsplänen und geschätzten Kosten für entwickelte SQL-Statements durchführen. Auf diese Weise gewonnene Erkenntnisse können vom Designer/Programmierer direkt genutzt werden, um Optimierungen in Hinblick auf die quasi grade entworfenen/implementierten SQL-Statements durchzuführen. Durch die gegebene zeitliche Nähe zum Anwendungs- und Datenbank-Design sind auch Performance-Optimierungen auf Basis von Datenmodellanpassungen (Normalisierung/Denormalisierung) ohne größeren Aufwand möglich.

Das beschriebene Vorgehen hat zwar den Vorteil, dass mögliche Performance-Probleme schon von den Akteuren (Designer/Programmierer) erkannt werden können, die diese durch Design-Änderungen am effektivsten zu lösen wissen. Demgegenüber erfordern die EXPLAIN-Analysen und das Verständnis der Ausführungspläne einen **Grad an Expertise**, den Designer/Programmierer in der Regel nicht besitzen. Ein Datenbank Administrator (DBA), der über diese verfügt, ist wiederum von den fachlichen Anforderungen zu distanziert, sodass er zwar mögliche Performance-Ausreißer erkennen, nicht aber fachlich bewerten kann. Führt eine Anwendung beispielsweise einmal monatlich eine sehr komplexe Auswertung mithilfe eines entsprechend Laufzeit-intensiven SQL-Statements durch, dann würde dem DBA diese Abfrage bei EXPLAIN-Analysen als kritisch erscheinen. Denn er weiß weder, dass damit ein fachlich aufwändiger Prozess durchgeführt wird, noch dass es sich dabei um eine einmalig pro Monat auszuführende Abfrage handelt. Um sich als DBA in einer Infrastruktur von nicht selten mehr als 100 unterschiedlichen Anwendungen über die fachlichen Anforderungen und speziellen Prozesse jeder einzelnen im Detail zu informieren beziehungsweise um sich als Designer/Programmierer das nötige Knowhow zur Ausführungsplanbewertung aufzubauen, ist personelle Kapazität vonnöten, die in der Regel nicht verfügbar ist.

Ein anderes Problem, dass sich in Zusammenhang mit frühzeitigen EXPLAIN-Analysen zeigt, begründet sich in dem dritten zuvor genannten Performance-Faktor: den Daten. Während diese bei Anwendungsweiterentwicklungen weitgehend vorliegen, existieren für neu zu entwickelnde Anwendungen im Normalfall keine **repräsentativen Datenbestände**. Somit fehlen auch geeignete Datenbankstatistiken zur Grundlage für die EXPLAIN-Auswertungen. Die Folge sind Ausführungspläne und Kostenabschätzungen, die mit denen eines späteren produktiven Einsatzes der Statements oftmals nur wenig gemeinsam haben und für eine proaktive Performance-Analyse somit (nahezu) unverwertbar sind.

Der im folgenden Kapitel vorgestellte proaktive modellbasierte Ansatz zur Performance-Analyse und -Vorhersage greift beide Probleme auf: die fehlende repräsentative Datenbasis für Datenbankstatistiken und die mangelnde Expertise zur Ausführungsplanbewertung durch Designer/Programmierer. Dabei sieht dieser Ansatz zur Bereitstellung geeigneter Datenbankstatistiken ein synthetisches Erzeugen anhand von Performance-Indikatoren vor. Das Problem der mangelnden Expertise wird durch eine einfache modellbasierte Darstellung von gewonnenen EXPLAIN-Ergebnissen adressiert. Wie diese gestaltet ist und mit den Performance-Indikatoren zusammenwirkt verdeutlichen die weiteren Ausführungen des Kapitels anhand *Abbildung 3*.

4.1 Performance-Indikatoren im Datenmodell

Als Performance-Indikatoren bezeichnet die vorliegende Arbeit ausgewählte **Metadaten zu Entitäten und deren Attributen** (beziehungsweise zu Tabellen und deren Spalten), die Aufschluss über die erwarteten realen Datenbestände geben und in Zusammenhang mit dem Datenbank-Design und der Infrastruktur erste Rückschlüsse auf die zukünftige Datenbank-Performance erlauben. Dazu zählen Informationen zu den erwarteten **Datenmengen** wie in etwa die erwartete Anzahl an Zeilen pro Tabelle und Kennzahlen zur **Datenverteilung** – beispielsweise in Form von Wertebereichsangaben, Einzelwertwahrscheinlichkeiten oder der Kardinalität pro Spalte. Viele dieser Informationen sind Teil des Ergebnisses der Anforderungsanalyse und somit frühzeitig im SDLC bekannt und vom Business Analyst erfasst worden. Dabei reicht die Dokumentation von rein textuellen Beschreibungen bis hin zu tief strukturierten Darstellungen. Eine einheitlich standardisierte Form zur Erfassung von Performance-Indikatoren im DBLC existiert jedoch bislang nicht, wodurch die Metadaten kaum bis gar nicht in den weiteren Entwicklungsprozess einfließen.

In der Praxis basiert die **Datenmodellierung** ähnlich wie weite Teile der Anwendungsmodellierung auf der Sprache UML. Dabei wurde diese ursprünglich nicht zur Abbildung von Datenstrukturen im Sinn einer Entity-Relationship-Modellierung konzipiert, sodass die Verbindung beider Welten – und damit die Modellierung von Anwendung und Datenstrukturen mithilfe einer gemeinsamen Sprache in einem gemeinsamen Tool – erst durch Ansätze wie [10] oder auch den Entwurf zum IMM Standard der OMG [11] geschaffen wurde. Die Voraussetzung dafür bildet jeweils die UML-Profil-Spezifikation, die es ermöglicht, bestehende UML-Objekte über Neu-Stereotypisierungen zu erweitern.

Um die zuvor genannten Performance-Indikatoren für den weiteren Entwicklungsprozess nutzbar zu machen und sie innerhalb bestehender Infrastrukturen/Tool-Landschaften standardisiert zu erfassen, kann ebenfalls der UML-Profil-Mechanismus genutzt werden. So ließe sich beispielsweise mithilfe eines geeigneten Profils wie in *Abbildung 3* in 1. schematisch angedeutet aus einem UML-Objekt „entity“ ein neues Objekt „entity_extended“ ableiten, das in einem zusätzlichen Merkmal „cardinality“ Informationen über die produktiv erwartete Datenmenge zu einer Entität/Tabelle aufnehmen kann.

4.2 Synthetische Datenbankstatistiken

Eines der eingangs aufgezeigten Hindernisse für proaktive Performance-Analysen beziehungsweise -Vorhersagen bestand in der **fehlenden repräsentativen Datenbasis** für Datenbank-Statistiken. Diese Statistiken werden im Normalfall vom DBMS anhand der gespeicherten Daten selbst gesammelt. Dem entgegen verfolgt das hier vorgestellte Konzept den Ansatz, dem DBMS Statistiken vorzugeben, ohne dazu datenbankseitig repräsentative Datenbestände vorhalten zu müssen. Dafür bieten zwar die wenigsten DBMS vordefinierte Schnittstellen an, allerdings sind sämtliche Statistik-Informationen in der Regel innerhalb DBMS-interner manipulierbarer Tabellen gespeichert, wie dies beispielswiese auch bei DB2 oder Oracle der Fall ist [12].

Datenbankstatistiken enthalten Informationen über **Datenmengen und Datenverteilungen** sowie Kennzahlen zur physischen Speicherung wie beispielsweise die Anzahl der verwendeten Datenbankseiten pro Tabelle. Während erstere inhaltlich den zuvor beschriebenen Performance-Indikatoren entsprechen, sind die Statistikdaten zur physischen Speicherung interne DBMS-abhängige Größen. Mithilfe geeigneter, von den DBMS-Herstellern zur Unterstützung beim Datenbank-Design bereitgestellter Abschätzungsvorschriften lassen sich aber auch diese Kennzahlen auf Grundlage der Performance-Indikatoren approximieren. Somit ist es wie in *Abbildung 3* in 2. gezeigt möglich, anhand geeignet formalisierter Performance-Indikatoren frühzeitig im SDLC/DBLC repräsentative Datenbankstatistiken künstlich zu erzeugen.

4.3 EXPLAIN und Performance-Modell

Auf Grundlage von synthetischen Datenbankstatistiken können wie in *Abbildung 3* in 3. und 4. zu sehen, mittels der vom DBMS bereitgestellten **EXPLAIN-Funktionalität**, der SQL-Workload und dem aus dem physischen Datenmodell ableitbaren Datenbankschema proaktive Performance-Vorhersagen durchgeführt werden. Die resultierenden, teils komplexen Ausführungspläne lassen sich allerdings nur mit ausreichend Expertise und vorhandenen personellen Kapazitäten angemessen auswerten, sodass diese Problematik vorläufig weiterbesteht. Eine Hauptursache, die das Verständnis von Ausführungsplänen erschwert, ist ihre hierarchische Darstellung als Zugriffsbaum. Demgegenüber denkt der Designer/Programmierer beim Modellieren oder dem Entwickeln von SQL-Statements auf relationale Weise. Die im vorliegenden Ansatz als Performance-Modell bezeichnete vereinfachte Präsentation von Ausführungsplänen versucht, diese Diskrepanz aufzulösen.

Das **Performance-Modell** basiert auf dem physischen Datenmodell und damit auf einer dem Designer/Programmierer bekannten Darstellungsform. Zusätzlich umfasst es die für diesen Personenkreis wesentlichen Informationen aus den EXPLAIN-Ergebnissen. Dazu zählen die vom DBMS abgeschätzten Kosten für die Ausführung des gesamten Statements sowie wichtiger Operatoren wie Tabellen- beziehungsweise Indexzugriffe oder Tabellenverknüpfungen mittels Join – jeweils skaliert um die erwartete Ausführungshäufigkeit des Statements. Weitere Detailinformationen innerhalb der Ausführungspläne wie beispielsweise die konkrete Abarbeitungsreihenfolge einzelner Operatoren oder Angaben zu abgeschätzten Prädikat-Selektivitäten werden vom Modell zum Zweck der Einfachheit und Verständlichkeit bewusst vernachlässigt. Für die gleichzeitige Analyse mehrerer Statements erfolgt eine Aggregation der jeweils abgeschätzten Kosten auf Objektebene.

Zentrale Komponente im Performance-Modell ist eine ebenfalls dem physischen Datenmodell angelehnte **Diagrammdarstellung**. Mithilfe farblicher Hervorhebung und geeigneter Bewertungsmetriken sollen sämtliche Objekte gemäß den vom DBMS geschätzten Zugriffskosten zur Abarbeitung der Workload klassifiziert und visualisiert werden. Auf diese Weise kann ein Designer/Programmierer frühzeitig Auskunft über aus Performance-Perspektive zu optimierende Bereiche im Datenbankschema beziehungsweise kritische, alternativ zu konzipierende SQL-Statements erhalten. *Abbildung 3* veranschaulicht exemplarisch ein visualisiertes Performance-Modell für zwei Statements/Ausführungspläne (EP). Während der untere Bereich weitgehend grün/unkritisch markiert ist, befinden sich im oberen Diagrammteil mögliche Performance-kritische rot gekennzeichnete Zugriffe, die es gezielt zu untersuchen und an geeigneter Stelle (SQL-Statement, Datenbank-Design) zu optimieren gilt (vgl. gestrichelte Pfeile in *Abbildung 3*).

Die **technische Realisierung** des Performance-Modells sowie der dazugehörigen Diagrammdarstellung erfolgt analog zur Erfassung der Performance-Indikatoren über den UML-Profil-Mechanismus, wodurch auch in diesem Punkt die Kompatibilität des vorgestellten Ansatzes zu bestehenden Tool-Infrastrukturen gewährleistet ist.

4.4 Ablauf einer Analyse/Vorhersage

Für den Designer/Programmierer sieht der in *Abbildung 3* vorgestellte proaktive Ansatz folgende Vorgehensweise vor. Nachdem nach 1. ein Datenbank-Design-Entwurf fertiggestellt ist, initiiert er in 2. einen Automatismus zur Abbildung des Designs in ein Datenbank-Schema sowie zur Erstellung von synthetischen Datenbank-Statistiken anhand der von ihm modellierten Performance-Indikatoren. Mithilfe einer weiteren Routine startet der Designer/Programmierer in 3. und 4. anschließend einen Simulationsprozess, der auf Basis der EXPLAIN-Mechanismen Performance-Vorhersagen für eine gegebene Workload erstellt und diese als Performance-Modell aufbereitet. Von dort aus informiert er sich mithilfe der Diagrammdarstellung über mögliche kritische Zugriffe, die er daraufhin gezielt analysiert und optimiert.

5. ZUSAMMENFASSUNG

Datenbank-Performance ist ein wichtiger, oftmals jedoch vernachlässigter Faktor in der Anwendungsentwicklung. Durch moderne Anforderungen und dazu implementierte Anwendungen sehen sich speziell deren Datenbank-Backends mit kontinuierlich wachsenden Herausforderungen insbesondere betreffend der Performance konfrontiert. Diese können nur bewältigt werden, wenn das Thema Datenbank-Performance intensiver betrachtet und durch proaktive Analysen (beispielsweise mittels EXPLAIN-Mechanismen) kontinuierlich verfolgt wird. Doch auch dann sind einzelne Hindernisse unvermeidlich: fehlende repräsentative Daten(-mengen) und Expertise/Kapazitäten zur Analyse.

Der vorliegende Beitrag präsentiert zur Lösung dieser Probleme einen modellbasierten Ansatz, der auf Basis synthetisch erzeugter Statistiken proaktive Performance-Analysen sowie -Vorhersagen erlaubt und die daraus gewonnenen Ergebnisse in einer einfach verständlichen Form visualisiert. Die technologische Grundlage dafür bietet die in der Praxis vorherrschende Modellierungssprache UML mit ihrer UML-Profil-Spezifikation. Sie erlaubt es das hier vorgestellte Konzept und die dazu benötigten Komponenten mit vorhandenen technischen Mitteln abzubilden und nahtlos in bestehende UML-Infrastrukturen zu integrieren.

6. AUSBLICK

Bei dem im Beitrag vorgestellten Konzept handelt es sich um einen auf Basis wiederkehrender praktischer Problemstellungen und den daraus gewonnenen Erfahrungen konstruierten Ansatz. Während die technische Umsetzbarkeit einzelner Teilaspekte wie etwa die Erfassung von Performance-Indikatoren oder die Konstruktion des Performance-Modells auf Basis von UML-Profilen bereits geprüft wurde, steht eine prototypische Implementierung des gesamten Prozesses zur Performance-Analyse noch aus.

Zuvor sind weitere Detailbetrachtungen nötig. So ist beispielsweise zu klären, in welchem Umfang **Performance-Indikatoren** im Datenmodell vom Analyst/Designer sinnvoll erfasst werden sollten. Dabei ist ein Kompromiss zwischen maximalem Detailgrad und minimal nötigem Informationsgehalt anzustreben, sodass der Aufwand zur Angabe von Performance-Indikatoren möglichst gering ist, mit deren Hilfe aber dennoch eine repräsentative Performance-Vorhersage ermöglicht wird.

Weiterhin gilt es, eine geeignete **Metrik zur Bewertung/Kategorisierung der Analyseergebnisse** zu entwickeln. Hier steht die Frage im Vordergrund, wann ein Zugriff anhand seiner Kosten als schlecht und wann er als gut zu bewerten ist. Ein teurer Zugriff ist nicht zwangsweise ein schlechter, wenn er beispielsweise zur Realisierung einer komplexen Funktionalität verwendet wird.

Zuletzt sei noch die Erfassung beziehungsweise Beschaffung der für die EXPLAIN-Analysen notwendigen **Workload** erwähnt. Diese muss dem vorgestellten proaktiven Analyseprozess zugänglich gemacht werden, um anhand des beschriebenen Konzepts frühzeitige Performance-Untersuchungen durchführen zu können. Im einfachsten Fall könnte angenommen werden, dass sämtliche SQL-Statements (inklusive ihrer Ausführungshäufigkeit) vom Designer/Programmierer ebenfalls im Datenmodell beispielsweise als zusätzliche Merkmale von Methoden in der UML-Klassenmodellierung zu erfassen und kontinuierlich zu pflegen wären. Dies wäre jedoch ein sehr aufwändiges Verfahren, das der gewünschten hohen Praxistauglichkeit des proaktiven Ansatzes entgegensteht. Somit sind alternative Varianten zur Beschaffung der Workload für den Analyseprozess zu untersuchen und abzuwägen.

7. LITERATUR

[1] C. Coronel, S. Morris, P. Rob. *Database Systems: Design, Implementation, and Management*, Course Technology, 10. Auflage, 2011.

[2] S. Salza, M. Renzetti. *A Modeling Tool for Workload Analysis and Performance Tuning of Parallel Database Applications*, Proceedings in ADBIS'97, 09.1997 http://www.bcs.org/upload/pdf/ewic_ad97_paper38.pdf

[3] R. Osman, W. J. Knottenbelt. *Database system performance evaluation models: A survey*, Artikel in Performance Evaluation, Elsevier Verlag, 10.2012 http://dx.doi.org/10.1016/j.peva.2012.05.006

[4] Tata Consultancy Services. *System and method for SQL performance assurance services*, Internationales Patent PCT/IN2011/000348, 11.2011 http://dx.doi.org/10.1016/j.peva.2012.05.006

[5] D. Wiese. Gewinnung, Verwaltung und Anwendung von Performance-Daten zur Unterstützung des autonomen Datenbank-Tuning, Dissertation, Fakultät für Mathematik und Informatik, Friedrich-Schiller-Universität Jena, 05.2011. http://www.informatik.uni-jena.de/dbis/alumni/wiese/pubs/Dissertation__David_Wiese.pdf

[6] S. Chaudhuri, V. Narasayya. *A Self-Tuning Database Systems: A Decade of Progress*, Proceedings in VLDB'07, 09.2007 http://research.microsoft.com/pubs/76506/vldb07-10yr.pdf

[7] N. Bruno, S. Chaudhuri. *An Online Approach to Physical Design Tuning*, Proceedings in ICDE'07, 04.2007 http://research.microsoft.com/pubs/74112/continuous.pdf

[8] Oracle Corporation. *Oracle Database 2 Day DBA 12c Release 1 (12.1) – Monitoring and Tuning the Database*, 2013. http://docs.oracle.com/cd/E16655_01/server.121/e17643/montune.htm#ADMQS103

[9] Microsoft Corporation. *SQL Server 2005 – Database Engine Tuning Advisor (DTA) in SQL Server 2005*, Technischer Artikel, 2006. http://download.microsoft.com/download/4/7/a/47a548b9-249e-484c-abd7-29f31282b04d/SQL2005DTA.doc

[10] C.-M. Lo. A Study of Applying a Model-Driven Approach to the Development of Database Applications, Dissertation, Department of Information Management, National Taiwan University of Science and Technology, 06.2012.

[11] Object Management Group. *Information Management Metamodel (IMM) Specification Draft Version 8.0*, Spezifikationsentwurf, 03.2009. http://www.omgwiki.org/imm/doku.php

[12] N. Burgold, M. Gerstmann, F. Leis. *Statistiken in relationalen DBMSen und Möglichkeiten zu deren synthetischer Erzeugung*, Projektarbeit, Fakultät für Mathematik und Informatik, Friedrich-Schiller-Universität Jena, 05.2014.

Big Data und der Fluch der Dimensionalität

Die effiziente Suche nach Quasi-Identifikatoren in hochdimensionalen Daten

Hannes Grunert
Lehrstuhl für Datenbank- und Informationssysteme
Universität Rostock
Albert-Einstein-Straße 22
hg(at)informatik.uni-rostock.de

Andreas Heuer
Lehrstuhl für Datenbank- und Informationssysteme
Universität Rostock
Albert-Einstein-Straße 22
ah(at)informatik.uni-rostock.de

Kurzfassung

In smarten Umgebungen werden häufig große Datenmengen durch eine Vielzahl von Sensoren erzeugt. In vielen Fällen werden dabei mehr Informationen generiert und verarbeitet als in Wirklichkeit vom Assistenzsystem benötigt wird. Dadurch lässt sich mehr über den Nutzer erfahren und sein Recht auf informationelle Selbstbestimmung ist verletzt.

Bestehende Methoden zur Sicherstellung der Privatheitsansprüche von Nutzern basieren auf dem Konzept sogenannter Quasi-Identifikatoren. Wie solche Quasi-Identifikatoren erkannt werden können, wurde in der bisherigen Forschung weitestgehend vernachlässigt.

In diesem Artikel stellen wir einen Algorithmus vor, der identifizierende Attributmengen schnell und vollständig erkennt. Die Evaluierung des Algorithmus erfolgt am Beispiel einer Datenbank mit personenbezogenen Informationen.

ACM Klassifikation

K.4.1 [**Computer and Society**]: Public Policy Issues—*Privacy*; H.2.4 [**Database Management**]: Systems—*Query Processing*

Stichworte

Datenbanken, Datenschutz, Big Data

1. EINLEITUNG

Assistenzsysteme sollen den Nutzer bei der Arbeit (Ambient Assisted Working) und in der Wohnung (Ambient Assisted Living) unterstützen. Durch verschiedene Sensoren werden Informationen über die momentane Situation und die Handlungen des Anwenders gesammelt. Diese Daten werden durch das System gespeichert und mit weiteren Daten, beispielsweise mit dem Facebook-Profil des Nutzers verknüpft. Durch die so gewonnenen Informationen lassen sich Vorlieben, Verhaltensmuster und zukünftige Ereignisse berechnen. Daraus werden die Intentionen und zukünftigen Handlungen des Benutzers abgeleitet, sodass die smarte Umgebung eigenständig auf die Bedürfnisse des Nutzers reagieren kann.

In Assistenzsystemen [17] werden häufig wesentlich mehr Informationen gesammelt als benötigt. Außerdem hat der Nutzer meist keinen oder nur einen sehr geringen Einfluss auf die Speicherung und Verarbeitung seiner personenbezogenen Daten. Dadurch ist sein Recht auf informationelle Selbstbestimmung verletzt. Durch eine Erweiterung des Assistenzsystems um eine Datenschutzkomponente, welche die Privatheitsansprüche des Nutzers gegen den Informationsbedarf des Systems überprüft, kann diese Problematik behoben werden.

Zwei Hauptaspekte des Datenschutzes sind Datenvermeidung und Datensparsamkeit. In §3a des Bundesdatenschutzgesetzes [1] wird gefordert, dass

> „[d]ie Erhebung, Verarbeitung und Nutzung personenbezogener Daten und die Auswahl und Gestaltung von Datenverarbeitungssystemen [...] an dem Ziel auszurichten [sind], so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen.“.

Mittels einer datensparsamen Weitergabe der Sensor- und Kontext-Informationen an die Analysewerkzeuge des Assistenzsystems wird nicht nur die Datenschutzfreundlichkeit des Systems verbessert. Bei der Vorverdichtung der Daten durch Selektion, Aggregation und Komprimierung am Sensor selbst lässt sich die Effizienz des Systems steigern. Die Privatheitsansprüche und der Informationsbedarf der Analysewerkzeuge können als Integritätsbedingungen im Datenbanksystem umgesetzt werden. Durch die Integritätsbedingungen lassen sich die notwendigen Algorithmen zur Anonymisierung und Vorverarbeitung direkt auf dem Datenbestand ausführen. Eine Übertragung in externe Programme bzw. Module, die sich evtl. auf anderen Recheneinheiten befinden, entfällt somit.

Für die Umsetzung von Datenschutzbestimmungen in smarten Umgebungen wird derzeit das PArADISE[1]-Framework entwickelt, welches insbesondere die Aspekte der Datensparsamkeit und Datenvermeidung in heterogenen Systemumgebungen realisieren soll.

In [3] stellen wir ein einfaches XML-Schema vor, mit der sich Privatheitsansprüche durch den Nutzer von smarten Systemen formulieren lassen. Dabei wird eine Anwendung

[1]Privacy-aware assistive distributed information system environment

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

innerhalb eines abgeschlossenen Systems in ihre Funktionalitäten aufgeteilt. Für jede Funktionalität lässt sich festlegen, welche Informationen in welchem Detailgrad an das System weitergegeben werden dürfen. Dazu lassen sich einzelne Attribute zu Attributkombinationen zusammenfassen, die angefragt werden können.

Für einen unerfahrenen Nutzer ist das Festlegen von sinnvollen Einstellungen nur schwer möglich. Die Frage, die sich ihm stellt, ist nicht die, ob er seine persönlichen Daten schützen soll, sondern vielmehr, welche Daten es wert sind, geschützt zu werden. Zur Kennzeichnung schützenswerter Daten werden u.a. sogenannte Quasi-Identifikatoren [2] verwendet. In diesem Artikel stellen wir einen neuen Ansatz vor, mit dem Quasi-Identifikatoren schnell und vollständig erkannt werden können.

Der Rest des Artikels ist wie folgt strukturiert: Kapitel 2 gibt einen aktuellen Überblick über den Stand der Forschung im Bereich der Erkennung von Quasi-Identifikatoren. Im folgenden Kapitel gehen wir detailliert darauf ein, wie schützenswerte Daten definiert sind und wie diese effizient erkannt werden können. Kapitel 4 evaluiert den Ansatz anhand eines Datensatzes. Das letzte Kapitel fasst den Beitrag zusammen und gibt einen Ausblick auf zukünftige Arbeiten.

2. STAND DER TECHNIK

In diesem Kapitel stellen wir bestehende Konzepte zur Ermittlung von Quasi-Identifikatoren (QI) vor. Außerdem werden Techniken vorgestellt, die in unseren Algorithmus eingefloßen sind.

2.1 Quasi-Identifikatoren

Zum Schutz personenbezogener Daten existieren Konzepte wie k-anonymity [16], l-diversity [8] und t-closeness [7]. Diese Konzepte unterteilen die Attribute einer Relation in Schlüssel, Quasi-Identifikatoren, sensitive Daten und sonstige Daten. Ziel ist es, dass die sensitiven Daten sich nicht eindeutig zu einer bestimmten Person zuordnen lassen. Da durch Schlüsselattribute Tupel eindeutig bestimmt werden können, dürfen diese unter keinen Umständen zusammen mit den sensitiven Attributen veröffentlicht werden.

Während Schlüssel im Laufe des Datenbankentwurfes festgelegt werden, lassen sich Quasi-Identifikatoren erst beim Vorliegen der Daten feststellen, da sie von den konkreten Attributwerten der Relation abhängen. Der Begriff Quasi-Identifikator wurde von Dalenius [2] geprägt und bezeichnet „a subset of attributes that can uniquely identify most tuples in a table".

Für „most tuples" wird häufig ein Grenzwert p festgelegt, der bestimmt, ob eine Attributkombination ein Quasi-Identifikator ist oder nicht. Dieser Grenzwert lässt sich beispielsweise in relationalen Datenbanken durch zwei SQL-Anfragen wie folgt bestimmen:

$$p = \frac{\text{COUNT DISTINCT * FROM (SELECT <attr-list> FROM table)}}{\text{COUNT * FROM table}} \qquad (1)$$

Wird für p der Wert 1 gewählt, so sind die gefundenen QI mit diesem Grenzwert auch Schlüssel der Relation. Um eine Vergleichbarkeit unseres Algorithmus mit dem von Motwani und Xu zu gewährleisten, verwenden wir ebenfalls die in (1) definierte „distinct ratio" (nach [12]).

Da es für den Ausdruck „die meisten" keinen standardisierten Quantor gibt, formulieren wir ihn mit dem Zeichen: $\overset{\geq p}{\forall}$, wobei p den Prozentsatz der eindeutig identifizierbaren Tupel (t_i) angibt. Ein Quasi-Identifikator $QI := \{A_1, ..., A_n\}$ ist für eine Relation R entsprechend definiert:

Quasi-Identifikator. $\overset{\geq p}{\forall}$ *$t_1, t_2 \in R$ [$t_1 \neq t_2 \Rightarrow \exists\ A \in QI$: $t_1(A) \neq t_2(A)$]*

Wie beim Datenbankentwurf reicht es auch für die Angabe von Quasi-Identifikatoren aus, wenn die minimale Menge von Attributen angegeben wird, welche die Eigenschaft eines QI hat. Eine solche Menge wird als minimaler Quasi-Identifikator bezeichnet.

minimaler Quasi-Identifikator. *X ist ein minimaler Quasi-Identifikator (mQI), wenn X ein Quasi-Identifikator ist und jede nicht-leere Teilmenge Y von X kein Quasi-Identifikator ist.*

X ist mQI: X ist QI $\wedge$ *($\nexists$ Y $\subset$ X: (Y $\neq \oslash$) $\wedge$ (Y ist QI))*

Insbesondere ist X kein minimaler Quasi-Identifikator, wenn eine Teilmenge X-{A} von X mit A $\in$ X existiert, die ein Quasi-Identifikator ist. Das Finden von allen Quasi-Identifikatoren stellt ein NP-vollständiges Problem dar, weil die Menge der zu untersuchenden Teilmengen exponentiell zur Anzahl der Attribute einer Relation steigt. Besteht eine Relation aus n Attributen, so existieren insgesamt 2^n Attributkombinationen, für die ermittelt werden muss, ob sie ein QI sind.

In [12] stellen Motwani und Xu einen Algorithmus zum effizienten Erkennen von minimalen Quasi-Identifikatoren vor. Dieser baut auf die von Mannila et. al [10] vorgeschlagene, ebenenweise Erzeugung von Attributmengen auf. Dabei wird die Minimalitätseigenschaft von Quasi-Identifikatoren sofort erkannt und der Suchraum beim Durchlauf auf der nächsten Ebene eingeschränkt.

Der Algorithmus ist effizienter als alle 2^n Teilmengen zu testen, allerdings stellt die von Big-Data-Anwendungen erzeugte Datenmenge eine neue Herausforderung dar. Insbesondere die hohe Dimensionalität und die Vielfalt der Daten sind ernst zu nehmende Probleme. Aus diesem Grund schlagen wir im folgenden Kapitel einen neuen Algorithmus vor, der auf den Algorithmus von Motwani und Xu aufsetzt.

2.2 Sideways Information Passing

Der von uns entwickelte Algorithmus verwendet Techniken, die bereits beim Sideways Information Passing (SIP, [4]) eingesetzt werden. Der grundlegende Ansatz von SIP besteht darin, dass während der Ausführung von Anfrageplänen Tupel nicht weiter betrachtet werden, sofern mit Sicherheit feststeht, dass sie keinen Bezug zu Tupeln aus anderen Relationen besitzen.

Durch das frühzeitige Erkennen solcher Tupel wird der zu betrachtende Suchraum eingeschränkt und die Ausführungszeit von Anfragen reduziert. Besonders effektiv ist dieses Vorgehen, wenn das Wissen über diese „magic sets" [14] zwischen den Teilen eines Anfrageplans ausgetauscht und in höheren Ebenen des Anfrageplans mit eingebunden wird. Beim SIP werden zudem weitere Techniken wie Bloomjoins [9] und Semi-Joins eingesetzt um den Anfrageplan weiter zu optimieren.

2.3 Effiziente Erfragung von identifizierenden Attributmengen

In [5] wird ein Algorithmus zur Ermittlung von identifizierenden Attributmengen (IA) in einer relationalen Datenbank beschrieben. Wird für eine Attributmenge erkannt, dass diese eine IA für eine Relation R ist, so sind auch alle Obermengen dieser Attributmenge IA für R. Ist für eine Relation bestehend aus den Attributen A, B und C bekannt, dass B eine identifizierende Attributmenge ist, dann sind auch AB, BC und ABC eine IA der Relation.

Ist eine Attributmenge hingegen keine IA für R, so sind auch alle Teilmengen dieser Attributmenge keine IA. Wenn beispielsweise AC keine IA für R ist, dann sind auch weder A noch C identifizierende Attributmengen für R. Attributmengen, die keine identifizierende Attributmenge sind, werden als *negierte Schlüssel* bezeichnet.

Der in [5] vorgestellte Algorithmus nutzt diese Eigenschaften um anhand eines Dialoges mit dem Nutzer die Schlüsseleigenschaften einer bereits existierenden Relation festzulegen. Dabei wird dem Nutzer ein Ausschnitt der Relationstabelle präsentiert anhand derer entschieden werden soll, ob eine Attributkombination Schlüssel ist oder nicht. Wird in einer Teilrelation festgestellt, dass die Attributmenge Tupel mit gleichen Attributwerten besitzt, so kann die Attributkombination für die Teilmenge, als auch für die gesamte Relation kein Schlüssel sein.

3. ALGORITHMUS

In diesem Kapitel stellen wir einen neuen Algorithmus zum Finden von minimalen Quasi-Identifikatoren vor. Der Algorithmus beschränkt sich dabei auf die Einschränkung der zu untersuchenden Attributkombinationen. Der entwickelte Ansatz führt dabei den von [12] vorgestellten Bottom-Up-Ansatz mit einen gegenläufigen Top-Down-Verfahren zusammen.

3.1 Bottom-Up

Der von Motwani und Xu in [12] vorgestellte Ansatz zum Erkennen aller Quasi-Identifikatoren innerhalb einer Relation nutzt einen in [10] präsentierten Algorithmus. Dabei wird für eine Relation mit n Attributen ebenenweise von den einelementigen zu n-elementigen Attributkombinationen Tests durchgeführt. Wird für eine i-elementige ($1 \leq i < n$) Attributkombination AK festgestellt, dass diese ein Quasi-Identifikator ist, so werden alle Attributkombinationen in den höheren Ebenen, die AK als Teilmenge enthalten, nicht mehr berücksichtigt.

Die Methodik ist in Algorithmus 1 kurz skizziert. Zunächst erfolgt eine Initialisierung der Datenbank. Dabei wird zudem die zu untersuchende Relation einmalig überprüft, um festzustellen, wie viele Tupel vorhanden sind. Anschließend werden alle einelementigen Attributmengen gebildet. Für jede Attributmenge wird überprüft, wie viele einzigartige Tupel in der Relation vorhanden sind und das Verhältnis zur Gesamtzahl an Tupeln gebildet. Liegt der Anteil über einen vorher bestimmten Grenzwert (*threshold*), so ist diese Attributmenge ein Quasi-Identifikator und wird in die Menge aller minimalen QIs *qiLowerSet* aufgenommen.

Sind alle Attributkombinationen überprüft, werden mittels Algorithmus 2 die nächstgrößeren Attributkombinationen unter Rücksichtnahme der bekannten QI gebildet. Die Überprüfung wird solange fortgesetzt, bis alle potentiellen Attributkombinationen überprüft worden sind.

Der Algorithmus arbeitet sehr effizient, da durch das Bottom-Up-Vorgehen die Minimalität der gefundenen QI sofort festgelegt ist. Besonders gut eignet sich der Algorithmus, wenn die Relation viele, aus wenigen Attributen zusammen-

Algorithm 1: bottomUp

```
Data: database table tbl, list of attributes elements
Result: a set with all minimal QI qiLowerSet
initialization();
for element in elements do
  | set := set ∪ {element}
end
while set is not empty do
  for Set testSet: set do
    double p := getPercentage(testSet, tbl);
    if $p \geq threshold$ then
      | qiLowerSet := qiLowerSet ∪ {testSet};
    end
  end
  set := buildNewLowerSet(set, elements);
end
return qiLowerSet;
```

Algorithm 2: buildNewLowerSet

```
Data: current lower set lSet, list of attributes
      elements
Result: the new lower set lSetNew
Set lSetNew := new Set();
for Set set: lSet do
  for Attribut A: elements do
    if $\nexists q \in qiLowerSet : q \subseteq set$ then
      | lSetNew := lSetNew ∪ {set ∪ {A}};
    end
  end
end
return lSetNew;
```

gesetzte QIs besitzt, da so der Suchraum gleich zu Beginn stark eingeschränkt wird.

Der Nachteil des Algorithmus zeigt sich, wenn die Relation QIs besitzt, die aus vielen Attributen zusammengesetzt sind. In diesem Fall wird der Suchraum erst zum Ende eingeschränkt, wodurch die Anzahl der zu betrachtenden Attributmengen nur unmerklich geringer ist. Falls die Relation sogar keine QIs besitzt, so erkennt der Algorithmus dies erst nach Überprüfung aller Kombinationen.

3.2 Top-Down

Für die Erklärung des Algorithmus 5 wird noch das zum Bottom-Up-Vorgehen entgegengesetzte Top-Down-Vorgehen benötigt. Dieses Verfahren setzt auf die in [5] vorgeschlagenen negierten Schlüssel auf. Analog zu negierten Schlüsseln gilt, dass eine Teilmenge T kein Quasi-Identifikator ist, wenn eine Attributkombination existiert, die kein QI ist und T als Teilmenge enthält.

Die Überprüfung der Attributkombinationen erfolgt wie beim Bottom-Up-Verfahren ebenenweise, jedoch in umgekehrter Reihenfolge. Für eine Relation mit n Attributen wird zunächst die n-elementige Teilmenge gebildet und geprüft. Anschließend werden alle (n-1)-elementigen Teilmengen gebildet. Dies wird solange fortgesetzt, bis alle einelementigen Teilmengen überprüft wurden. Die gefundenen Quasi-Identifikatoren werden in *qiUpperSet* gespeichert.

Durch das Top-Down-Vorgehen ist die Minimalität der Quasi-Identifikatoren nicht gewährleistet. Dieser Nachteil

Algorithm 3: buildNewUpperSet

```
Data: current upper set uSet
Result: the new upper set uSetNew
Set uSetNew := new Set();
for Set set: uSet do
    for Attribut A: set do
        if $\nexists o \in$ optOutSet: set - $\{A\} \subseteq o$ then
            uSetNew := uSetNew $\cup$ {set - {A}};
        end
    end
end
return uSetNew;
```

lässt sich dadurch beheben, dass wenn auf Ebene k ein QI gefunden wird, die entsprechenden Obermengen in den Ebenen mit mehr als k Attributen gestrichen werden. In den Algorithmen 3 und 4 ist das Top-Down-Vorgehen skizziert.

Algorithm 4: topDown

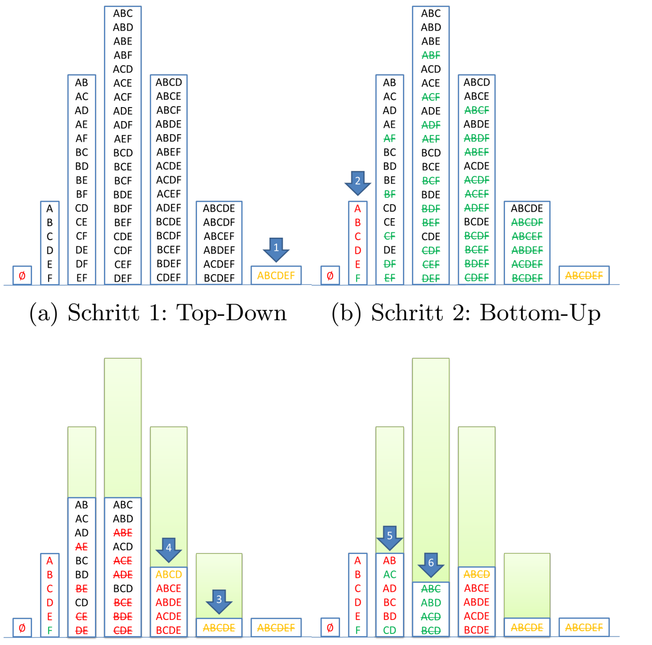
```
Data: database table tbl, list of attributes elements
Result: a set with all minimal quasi-identifier qiSet
initialization();
set := elements;
Set optOutSet := new Set();
Set qiUpperSet := new Set();
while set is not empty do
    for Set<String> testSet: set do
        double p := getPercentage(testSet, tbl);
        if $p <$ threshold then
            optOutSet := optOutSet $\cup$ {subset};
        else
            qiUpperSet := qiUpperSet $\cup$ {testSet};
            for Set o: qiSet do
                if testSet $\subset$ o then
                    qiUpperSet := qiUpperSet - {o};
                end
            end
        end
    end
    set := buildNewUpper(set);
end
return qiUpperSet;
```

Der Top-Down-Ansatz hebt die Nachteile des Bottom-Up-Vorgehens auf: der Algorithmus arbeitet effizient, wenn QIs aus vielen Attributen zusammengesetzt sind und für den Fall, dass die gesamte Relation kein QI ist, wird dies bei der ersten Überprüfung erkannt und der Algorithmus terminiert dann umgehend.

Besteht die Relation hingegen aus vielen kleinen QIs, dann wird der Suchraum erst zum Ende des Algorithmus stark eingeschränkt. Ein weiterer Nachteil liegt in der erhöhten Rechenzeit, auf die in der Evaluation näher eingegangen wird.

3.3 Bottom-Up+Top-Down

Der in diesem Artikel vorgeschlagene Algorithmus kombiniert die oben vorgestellten Verfahren. Dabei werden die Verfahren im Wechsel angewandt und das Wissen über (negierte) Quasi-Identifikatoren wie beim Sideways Information


(a) Schritt 1: Top-Down (b) Schritt 2: Bottom-Up

(c) Schritt 3+4: Top-Down (d) Schritt 5+6: Bottom-Up

Abbildung 1: Veranschaulichung des Bottom-Up+Top-Down-Algorithmus

Passing [4] untereinander ausgetauscht. Es wird pro Berechnungsschritt entweder die Top-Down- oder die Bottom-Up-Methode angewandt und das Ergebnis an die jeweils andere Methode übergeben. Der Algorithmus terminiert, sobald alle Attributebenen durch einen der beiden Methoden abgearbeitet wurden oder das Bottom-Up-Vorgehen keine Attributkombinationen mehr zu überprüfen hat. In Abbildung 1 ist die Arbeitsweise des Algorithmus anhand einer Beispielrelation mit sechs Attributen dargestellt. Die rot markierten Kombinationen stehen dabei für negierte QI, grün markierte für minimale QI und gelb markierte für potentiell minimale QI.

Um zu entscheiden, welcher Algorithmus im nächsten Zyklus angewandt wird, wird eine Wichtungsfunktion eingeführt. Die Überprüfung einer einzelnen Attributkombination auf Duplikate hat eine Laufzeit von O(n*log(n)), wobei n die Anzahl der Tupel in der Relation ist. Die Überprüfung der Tupel hängt aber auch von der Größe der Attributkombination ab. Besteht ein zu überprüfendes Tupel aus mehreren Attributen, so müssen im Datenbanksystem auch mehr Daten in den Arbeitsspeicher für die Duplikaterkennung geladen werden. Durch große Datenmengen werden Seiten schnell aus dem Arbeitsspeicher verdrängt, obwohl sie später wieder benötigt werden. Dadurch steigt die Rechenzeit weiter an.

Für eine vereinfachte Wichtungsfunktion nehmen wir an, dass alle Attribute den gleichen Speicherplatz belegen. Die Anzahl der Attribute in einer Attributkombination bezeichnen wir mit *m*. Für die Duplikaterkennung ergibt sich dann eine Laufzeit von O((n*m)*log(n*m)).

Da die Anzahl der Tupel für jede Duplikaterkennung konstant bleibt, kann n aus der Kostenabschätzung entfernt werden. Die Kosten für die Überprüfung einer einzelnen

Algorithm 5: bottomUpTopDown

```
Data: database table tbl, list of attributes attrList
Result: a set with all minimal quasi-identifier qiSet
attrList.removeConstantAttributes();
Set upperSet := new Set({attrList});
Set lowerSet := new Set(attrList);
// Sets to check for each algorithm
int bottom := 0;
int top := attrList.size();
while (bottom<=top) or (lowerSet is empty) do
    calculateWeights();
    if isLowerSetNext then
        bottomUp();
        buildNewLowerSet();
        bottom++;
        // Remove new QI from upper set
        modifyUpperSet();
    else
        topDown();
        buildNewUpperSet();
        top--;
        // Remove new negated QI from lower set
        modifyLowerSet();
    end
end
qiSet := qiLowerSet ∪ qiUpperSet;
return qiSet;
```

Attributkombination mit m Attributen beträgt demnach O((m*log(m)).

Die Gesamtkosten für das Überprüfen der möglichen Quasi-Identifikatoren werden mit W_{AVG} bezeichnet. W_{AVG} ergibt sich aus dem Produkt für das Überprüfen einer einzelnen Attributkombination und der Anzahl der Attributkombinationen ($AttrK_n$) mit n Attributen.

$$W_{AVG} := AttrK_n * log(m) * m \qquad (2)$$

Soll die Wichtungsfunktion präziser sein, so lässt sich der Aufwand abschätzen, indem für jede Attributkombination X die Summe s über die Attributgrößen von X gebildet und anschließend gewichtet wird. Die Einzelgewichte werden anschließend zum Gesamtgewicht aufsummiert.

$$W_{AVG} := \sum_{X \in AttrK_n} log(s) * s; \; s = \sum_{A \in X} size(A) \qquad (3)$$

Diese Wichtung eignet sich allerdings nur, wenn Zugang zu den Metadaten der Datenbankrelation besteht.

4. EVALUATION

Für die Evaluation des Algorithmus wurde die „Adult“-Relation aus dem UCI Machine Learning Repository [6] verwendet. Die Relation besteht aus anonymisierten, personenbezogenen Daten, bei denen Schlüssel sowie Vor- und Nachname von Personen entfernt wurden. Die übrigen 15 Attribute enthalten Angaben zu Alter, Ehestand, Staatsangehörigkeit und Schulabschluss. Die Relation besteht insgesamt aus 32561 Tupeln, die zunächst im CSV-Format vorlagen und in eine Datenbank geparst wurden.

Die Evaluation erfolgte in einer Client-Server-Umgebung. Als Server dient eine virtuelle Maschine, die mit einer 64-Bit-CPU (vier Kerne @ 2 GHz und jeweils 4 MB Cache) und 4 GB Arbeitsspeicher ausgestattet ist. Auf dieser wurde eine MySQL-Datenbank mit InnoDB als Speichersystem verwendet. Der Client wurde mit einem i7-3630QM als CPU betrieben. Dieser bestand ebenfalls aus vier Kernen, die jeweils über 2,3 GHz und 6 MB Cache verfügten. Als Arbeitsspeicher standen 8 GB zur Verfügung. Als Laufzeitumgebung wurde Java SE 8u5 eingesetzt.

Der Datensatz wurde mit jedem Algorithmus getestet. Um zu ermitteln, wie die Algorithmen sich bei verschiedenen Grenzwerten für Quasi-Identifikatoren verhalten, wurden die Tests mit 10 Grenzwerten zwischen 50% und 99% wiederholt.

Die Tests mit den Top-Down- und Bottom-Up-Algorithmen benötigten im Schnitt gleich viele Tablescans (siehe Abbildung 2). Die Top-Down-Methode lieferte bessere Ergebnisse bei hohen QI-Grenzwerten, Bottom-Up ist besser bei niedrigeren Grenzwerten. Bei der Laufzeit (siehe Abbildung 3) liegt die Bottom-Up-Methode deutlich vor dem Top-Down-Ansatz. Grund hierfür sind die großen Attributkombinationen, die der Top-Down-Algorithmus zu Beginn überprüfen muss.

Der Bottom-Up+Top-Down-Ansatz liegt hinsichtlich Laufzeit als auch bei der Anzahl der Attributvergleiche deutlich vorne. Die Anzahl der Tablescans konnte im Vergleich zum Bottom-Up-Verfahren zwischen 67,4% (4076 statt 12501 Scans; Grenzwert: 0.5) und 96,8% (543 statt 16818 Scans; Grenzwert 0.9) reduziert werden. Gleiches gilt für die Laufzeit (58,1% bis 97,5%; siehe Abbildung 3).

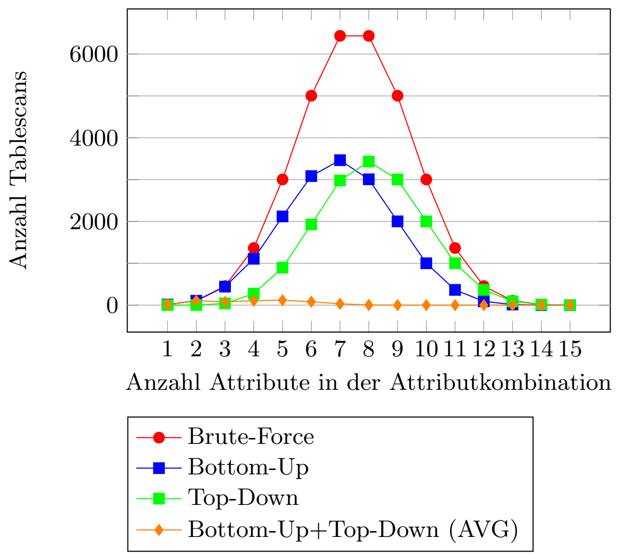


Abbildung 2: Verhältnis von der Anzahl der Attribute in den Attributkombinationen zur Anzahl von Tablescans (Adult-DB, Grenzwert 90%)

Wie in Abbildung 3 zu erkennen ist, nimmt die Laufzeit beim Bottom-Up+Top-Down-Verfahren im Grenzwertbereich von 70%-90% stark ab. Interessant ist dies aus zwei Gründen. Erstens nimmt die Anzahl der Quasi-Identifikatoren bis 90% ebenfalls ab (179 bei 50%, 56 bei 90%). Dies legt nahe, dass die Skalierung des Verfahrens neben der Dimension der Relation (Anzahl von Tupel und

Attributen) auch von der Anzahl der vorhandenen QIs abhängt. Um den Zusammenhang zu bestätigen, sind aber weitere Untersuchungen erforderlich.

Zweitens wird dieser Grenzwertbereich in der Literatur [13] häufig benutzt, um besonders schützenswerte Daten hervorzuheben. Durch die gute Skalierung des Algorithmus in diesem Bereich lassen sich diese QIs schnell feststellen.

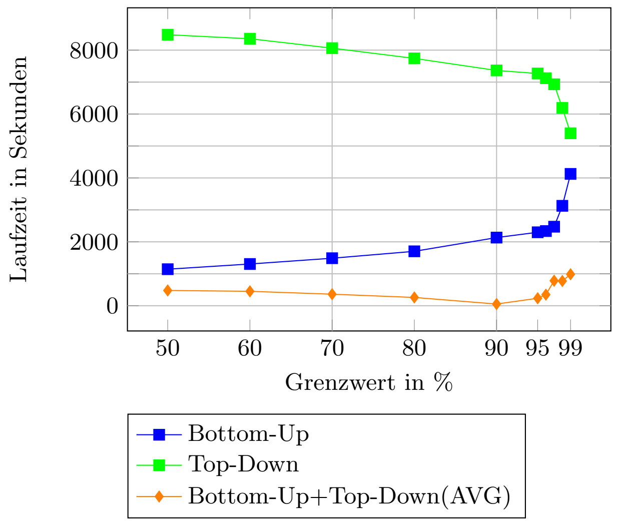


Abbildung 3: Vergleich der Laufzeit der verschiedenen Algorithmen (Adult-DB)

5. AUSBLICK

In dieser Arbeit stellten wir einen effizienten Algorithmus zur Erkennung von QI in hochdimensionalen Daten vor. Anhand eines Beispiels mit Sensordaten zeigten wir die Eignung in Assistenzsystemen. Darüber hinaus ermitteln wir derzeit, inwiefern sich QIs in temporalen Datenbanken feststellen lassen. Das so gewonnene Wissen über schützenswerte Daten wird in unser Gesamtprojekt zur datenschutzfreundlichen Anfrageverarbeitung in Assistenzsystemen eingebunden.

In späteren Untersuchungen werden wir testen, welche weiteren Quasi-Identifikatoren sich aus der Kombination von Daten verschiedener Relationen ableiten lassen. Der dafür verwendete Datensatz besteht aus Sensordaten, die im Smart Appliance Lab des Graduiertenkollegs MuSAMA durch ein Tool [11] aufgezeichnet wurden. Die Daten umfassen dabei Bewegungsprofile, die mittels RFID-Tags und einen Sensfloor [15] erfasst wurden, aber auch Informationen zu Licht und Temperatur. Eine Verknüpfung der Basis-Relationen erfolgt dabei über die ermittelten Quasi-Identifikatoren.

6. DANKSAGUNG

Hannes Grunert wird durch die Deutsche Forschungsgemeinschaft (DFG) im Rahmen des Graduiertenkollegs 1424 (Multimodal Smart Appliance Ensembles for Mobile Applications - MuSAMA) gefördert. Wir danken den anonymen Gutachtern für ihre Anregungen und Kommentare.

7. LITERATUR

[1] Bundesrepublik Deutschland. Bundesdatenschutzgesetz in der Fassung der Bekanntmachung vom 14. Januar 2003, das zuletzt durch Artikel 1 des Gesetzes vom 14. August 2009 geändert worden ist, 2010.

[2] T. Dalenius. Finding a Needle In a Haystack or Identifying Anonymous Census Records. *Journal of Official Statistics*, 2(3):329–336, 1986.

[3] H. Grunert. Privacy Policy for Smart Environments. `http://www.ls-dbis.de/pp4se`, 2014. zuletzt aufgerufen am 17.07.2014.

[4] Z. G. Ives and N. E. Taylor. Sideways information passing for push-style query processing. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 774–783. IEEE, 2008.

[5] M. Klettke. *Akquisition von Integritätsbedingungen in Datenbanken*. PhD thesis, Universität Rostock, 1997.

[6] R. Kohavi and B. Becker. Adult Data Set. `http://archive.ics.uci.edu/ml/datasets/Adult`, 1996. zuletzt aufgerufen am 17.07.2014.

[7] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *ICDE*, volume 7, pages 106–115, 2007.

[8] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[9] L. F. Mackert. R* optimizer validation and performance evaluation for distributed queries. In *Readings in database systems*, pages 219–229. Morgan Kaufmann Publishers Inc., 1988.

[10] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[11] D. Moos. Konzepte und Lösungen für Datenaufzeichnungen in heterogenen dynamischen Umgebungen. Bachelorarbeit, Universität Rostock, 2011.

[12] R. Motwani and Y. Xu. Efficient algorithms for masking and finding quasi-identifiers. In *Proceedings of the Conference on Very Large Data Bases (VLDB)*, pages 83–93, 2007.

[13] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, Technical report, SRI International, 1998.

[14] P. Seshadri, J. M. Hellerstein, H. Pirahesh, T. Leung, R. Ramakrishnan, D. Srivastava, P. J. Stuckey, and S. Sudarshan. Cost-based optimization for magic: Algebra and implementation. In *ACM SIGMOD Record*, volume 25, pages 435–446. ACM, 1996.

[15] A. Steinhage and C. Lauterbach. Sensfloor (r): Ein AAL Sensorsystem für Sicherheit, Homecare und Komfort. *Ambient Assisted Living-AAL*, 2008.

[16] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[17] M. Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.

Combining Spotify and Twitter Data for Generating a Recent and Public Dataset for Music Recommendation

Martin Pichl
Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
martin.pichl@uibk.ac.at

Eva Zangerle
Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
eva.zangerle@uibk.ac.at

Günther Specht
Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
guenther.specht@uibk.ac.at

ABSTRACT

In this paper, we present a dataset based on publicly available information. It contains listening histories of Spotify users, who posted what they are listening at the moment on the micro blogging platform Twitter. The dataset was derived using the Twitter Streaming API and is updated regularly. To show an application of this dataset, we implement and evaluate a pure collaborative filtering based recommender system. The performance of this system can be seen as a baseline approach for evaluating further, more sophisticated recommendation approaches. These approaches will be implemented and benchmarked against our baseline approach in future works.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.2.8 [**Database Applications**]: Data mining

General Terms

Algorithms, Experimentation

Keywords

Music Recommender Systems, Collaborative Filtering, Social Media

1. INTRODUCTION

More and more music is available to be consumed, due to new distribution channels enabled by the rise of the web. Those new distribution channels, for instance music streaming platforms, generate and store valuable data about users and their listening behavior. However, most of the time the data gathered by these companies is not publicly available. There are datasets available based on such private data corpora, which are widely used for implementing and evaluating recommender systems, i.e., the million song dataset (MSD) [4], however such datasets like the MSD often are not recent anymore. Thus, in order to address the problem of a lack of recent and public available data for the development and evaluation of recommender systems, we exploit the fact that many users of music streaming platforms post what they are listening to on the microblogging Twitter. An example for such a tweet is "#NowPlaying Human (The Killers) #craigcardiff #spotify http://t.co/N08f2MsdSt". Using a dataset derived from such tweets, we implement and evaluate a collaborative filtering (CF) based music recommender system and show that this is a promising approach. Music recommender systems are of interest, as the volume and variety of available music increased dramatically, as mentioned in the beginning. Besides commercial vendors like Spotify[1], there are also open platforms like SoundCloud[2] or Promo DJ[3], which foster this development. On those platforms, users can upload and publish their own creations. As more and more music is available to be consumed, it gets difficult for the user or rather customer to navigate through it. By giving music recommendations, recommender systems help the user to identify music he or she wants to listen to without browsing through the whole collection. By supporting the user finding items he or she likes, the platform operators benefit from an increased usability and thus increase the customer satisfaction.

As the recommender system implemented in this work delivers suitable results, we will gradually enlarge the dataset by further sources and assess how the enlargements influences the performance of the recommender system in future work. Additionally, as the dataset also contains time stamps and a part of the captured tweets contains a geolocation, more sophisticated recommendation approaches utilizing these additional context based information can be compared against the baseline pure CF-based approach in future works.

The remainder of this paper is structured as follows: in Section 2 we present the dataset creation process as well as the dataset itself in more detail. Afterwards, in Section 3 we briefly present the recommendation approach, which is evaluated in Section 4. Before we present the conclusion drawn from the evaluation on Section 6, related work is discussed in Section 5.

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

[1]`http://www.spotify.com`
[2]`http://soundcloud.com`
[3]`http://promodj.com`

2. THE SPOTIFY DATASET

In this Section, the used dataset [4] for developing and evaluating the recommender system is presented.

2.1 Dataset Creation

For the crawling of a sufficiently large dataset, we relied on the Twitter Streaming API which allows for crawling tweets containing specified keywords. Since July 2011, we crawled for tweets containing the keywords *nowplaying, listento* and *listeningto*. Until October 2014, we were able to crawl more than 90 million tweets. In contrast to other contributions aiming at extracting music information from Twitter, where the tweet's content is used to extract artist and track information from [17, 7, 16], we propose to exploit the subset of crawled tweets containing a URL leading to the website of the Spotify music streaming service[5]. I.e., information about the artist and the track are extracted from the website mentioned in the tweet, rather than from the content of the tweet. This enables us an unambiguous resolution of the tweets, in contradiction to the contributions mentioned above, where the text of the tweets is compared to entries in the reference database using some similarity measure. A typical tweet, published via Spotify, is depicted in the following: "#nowPlaying I Tried by Total on #Spotify http://t.co/ZaFH ZAokbV", where a user published that he or she listened to the track "I Tried" by the band "Total" on Spotify. Additionally, a shortened URL is provided. Besides this shortened URL, Twitter also provides the according resolved URL via its API. This allows for directly identifying all Spotify-URLs by searching for all URLs containing the string "spotify.com" or "spoti.fi". By following the identified URLs, the artist and the track can be extracted from the title tag of the according website. For instance, the title of the website behind the URL stated above is "<title>I tried by Total on Spotify </title>". Using the regular expression "<title>(.*) by (.*) on.*</title>" the name of the track (group 1) and the artist (group 2) can be extracted.

By applying the presented approach to the crawled tweets, we were able to extract artist and track information from 7.08% of all tweets or rather 49.45% of all tweets containing at least one URL. We refer to the subset of tweets, for which we are able to extract the artist and the track, as "matched tweets". An overview of the captured tweets is given in Table 1. 1.94% of the tweets containing a Spotify-URL couldn't be matched due to HTTP 404 Not Found and HTTP 500 Internal Server errors.

Restriction	Number of Tweets	Percentage
None	90,642,123	100.00%
At least one URL	12,971,482	14.31%
A Spotify-URL	6,541,595	7.22%
Matched	6,414,702	7.08%

Table 1: Captured and Matched Tweets

Facilitating the dataset creation approach previously presented, we are able to gather 6,414,702 tweets and extract artist and track data from the contained Spotify-URLs.

[4]available at: `http://dbis-twitterdata.uibk.ac.at/spotifyDataset/`

[5]`http://www.spotify.com`

2.2 Dataset Description

Based on the raw data presented in the previous Section, we generate a final dataset of <user, artist, track>-triples which contains 5,504,496 tweets of 569,722 unique users who listened to 322,647 tracks by 69,271 artists. In this final dataset, users considered as not valuable for recommendations, i.e., the @SpotifyNowPlay Twitter account which retweets tweets sent via @Spotify, are removed. These users were identified manually by the authors.

As typical for social media datasets, our dataset has a long-tailed distribution among the users and their respective number of posted tweets [5]. This means that there are only a few number of users tweeting rather often in this dataset and numerous users are tweeted rarely which can be found in the long-tail. This long-tailed distribution can be seen in Table 2 and Figure 1, where the logarithm of the number of tweets is plotted against the corresponding number of users.

Number of Tweets	Number of Users
>0	569,722
>1	354,969
>10	91,217
>100	7,419
>1,000	198

Table 2: Number of Tweets and Number of Users

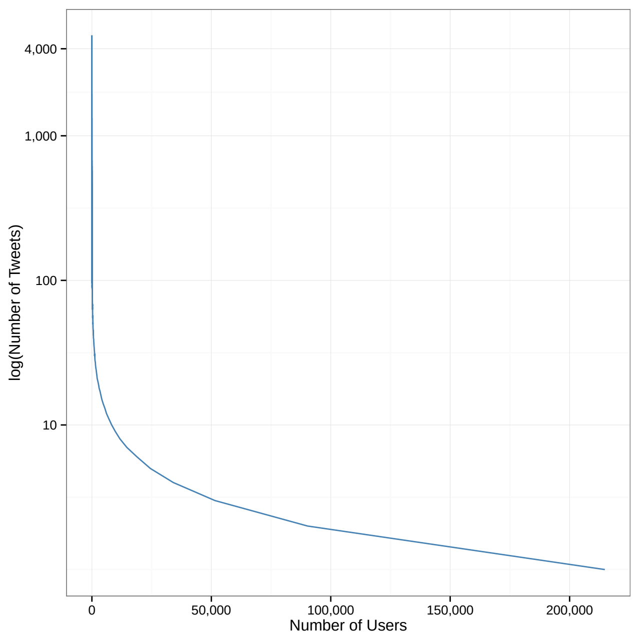


Figure 1: Number of Tweets versus Number of Users

The performance of a pure collaborative filtering-based recommender system increases with the detailedness of a user profile. Especially for new users in a system, where no or only little data is available about them, this poses a problem as no suitable recommendations can be computed. In our case, problematic users are users who tweeted rarely and thus can be found in the long tail.

Besides the long-tail among the number of posted tweets, there is another long-tail among the distribution of the artist play-counts in the dataset: there are a few popular artists occurring in a large number of tweets and many artists are mentioned only in a limited number of tweets. This is shown in Figure 2, where the logarithm of the number of tweets in which an artist occurs in (the play-count) is plotted against the number of artists. Thus, this plot states how many artists are mentioned how often in the dataset.

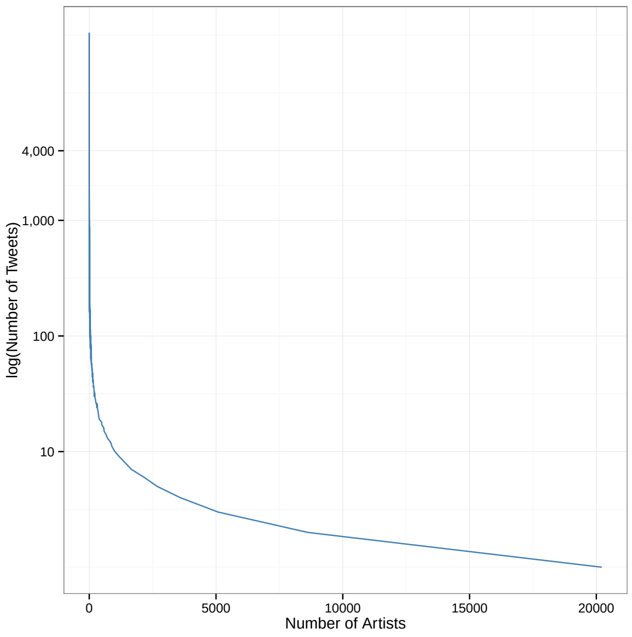


Figure 2: Play-Count versus Number of Artists

How the presented dataset is used as input- and evaluation data for a music recommender system, is presented in the next Section.

3. THE BASELINE RECOMMENDATION APPROACH

In order to present how the dataset can be applied, we use our dataset as input and evaluation data for an artist recommendation system. This recommender system is based on the open source machine learning library Mahout[2]. The performance of this recommender system is shown in Section 4 and serves as a benchmark for future work.

3.1 Recommendation Approach

For showing the usefulness of our dataset, we implemented a User-based CF approach. User-based CF recommends items by solely utilizing past user-item interactions. For the music recommender system, a user-item interaction states that a user listened to a certain track by a certain artist. Thus, the past user-item interactions represent the listening history of a user. In the following, we describe our basic approach taken for computing artist recommendations and provide details about the implementation.

In order to estimate the similarity of two users, we computed a linear combination of the Jaccard-Coefficients [10] based on the listening histories of the user. The Jaccard-Coefficient is defined in Equation 1 and measures the proportion of common items in two sets.

$$jaccard_{i,j} = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \tag{1}$$

For each user, there are two listening histories we take into consideration: the set of all tracks a user listened to and the set of all artists a user listened to. Thus, we are able to compute a *artist similartiy (artistSim)* and a *track similarity (trackSim)* as shown in Equations 2 and 3.

$$artistSim_{i,j} = \frac{|artists_i \cap artists_j|}{|artists_i \cup artists_j|} \tag{2}$$

$$trackSim_{i,j} = \frac{|tracks_i \cap tracks_j|}{|tracks_i \cup tracks_j|} \tag{3}$$

The final user similarity is computed using a weighted average of both, the *artistSim* and *trackSim* as depicted in Equation 4.

$$sim_{i,j} = w_a * artistSim_{i,j} + w_t * trackSim_{i,j} \tag{4}$$

The weights w_a and w_t determine the influence of the artist- and the track listening history on the user similarity, where $w_a + w_t = 1$. Thus, if $w_t = 0$, only the artist listening history is taken into consideration. We call such a recommender system an artist-based recommender system. Analogously, if $w_a = 0$ we call such a recommender system track-based. If $w_a > 0 \wedge w_t > 0$, both the artist- and track listening histories are used. Hence, we facilitate a hybrid recommender system for artist recommendations.

The presented weights have to be predetermined. In this work, we use a grid-search for finding suitable input parameter for our recommender system as described in Section 4.2.

4. EVALUATION

In this Section we present the performance of the implemented artist recommender system, but also discuss the limitations of the conducted offline evaluation.

4.1 Evaluation Setup

The performance of the recommender system with different input parameters was evaluated using *precision* and *recall*. Although we focus on the *precision*, for the sake of completeness we also include the *recall* into the evaluation, as this is usual in the field of information retrieval [3]. The metrics were computed using a Leave-n-Out algorithm, which can be described as follows:

1. Randomly remove n items from the listening history of a user
2. Recommend m items to the user
3. Calculate *precision* and *recall* by comparing the m recommended and the n removed items
4. Repeat step 1 to 3 p times
5. Calculate the mean *precision* and the mean *recall*

Each evaluation in the following Sections has been repeated five times ($p = 5$) and the size of the test set was fixed to 10 items ($r = 10$). Thus, we can evaluate the performance of the recommender for recommending up to 10 items.

4.2 Determining the Input Parameters

In order to determine good input parameters for the recommender system, a grid search was conducted. Therefore, we define a grid of parameters and the possible combinations are evaluated using a performance measure [9]. In our case, we relied on the *precision* of the recommender system (cf. Figure 3), as the task of a music recommender system is to find a certain number of items a user will listen to (or buy), but not necessarily to find all good items. *Precision* is a reasonable metric for this so called *Find Good Items* task [8] and was assessed using the explained Leave-n-Out algorithm. For this grid search, we recommended one item and the size of the test set was fixed to 10 items. In order to find good input parameters, the following grid parameters determining the computation of the user similarity were altered:

- Number of nearest neighbors k
- Weight of the artist similarity w_a
- Weight of the track similarity w_t

The result can be seen in Figure 3. For our dataset it holds, that the best results are achieved with a track-based recommender system ($w_a = 0$,$w_t = 1$) and 80 nearest neighbors ($k = 80$). Thus, for the performance evaluation of the recommender system in the next Section, we use the following parameters:

- Number of nearest neighbors 80
- Weight of the artist similarity 0
- Weight of the track similarity 1

4.3 Performance of the Baseline Recommender System

In this Section, the performance of the recommender system using the optimized input parameters is presented. Prior to the evaluation, we also examined real implementations of music recommender systems: Last.fm, a music discovery service, for instance recommends 6 artists[6] when displaying a certain artist. If an artist is displayed on Spotify[7], 7 similar artists are recommended at the first page. This number of items also corresponds to the work of Miller [11], who argues that people are able to process about 7 items at a glance, or rather that the span of attention is too short for processing long lists of items. The *precision@6* and the *precision@7* of our recommender are 0.20 and 0.19, respectively. In such a setting, 20% of the recommended items computed by the proposed recommender system would be a hit. In other words, a customer should be interested in at least in two of the recommended artists. An overview about the *precision@n* of the recommender is given in Table 3.

[6]http://www.last.fm/music/Lana+Del+Rey
[7]http://play.spotify.com/artist/00FQb4jTyendYWaN8pK0wa

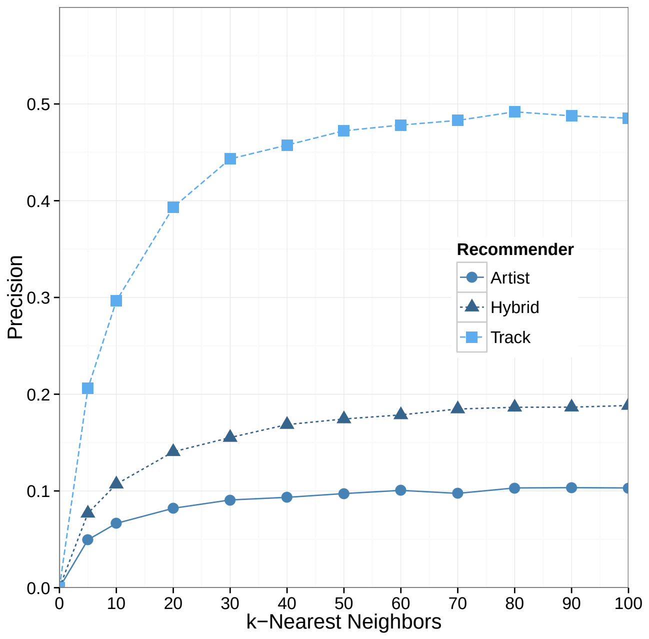


Figure 3: Precision and Recall of the Track-Based Recommender

n	Precision	Recall	Upper Bound
1	0.49	0.05	0.10
5	0.23	0.11	0.50
6	0.20	0.12	0.60
7	0.19	0.13	0.70
10	0.15	0.15	1.00

Table 3: Precision and Recall of the Track-Based Recommender

As shown in Figure 4, with an increasing number of recommendations, the performance of the presented recommender system declines. Thus, for a high number of recommendations the recommender system is rather limited. This is, as the chance of false positives increases if the size of the test set is kept constant. For computing the *recall* metric, the 10 items in the test set are considered as relevant items (and hence are desirable to recommend to the user). The recall metric describes the fraction of relevant artists who are recommended, i.e., when recommending 5 items, even if all items are considered relevant, the maximum recall is still only 50% as 10 items are considered as relevant. Thus, in the evaluation setup, recall is bound by an upper limit, which is the number of recommended items divided by the size of the test set.

4.4 Limitations of the Evaluation

Beside discussing the results, it is worth to mention also two limitations in the evaluation approach: First, only recommendations for items the user already interacted with can be evaluated [5]. If something new is recommended, it can't be stated whether the user likes the item or not. We can only state that it is not part of the user's listening history in our dataset. Thus, this evaluation doesn't fit to the per-

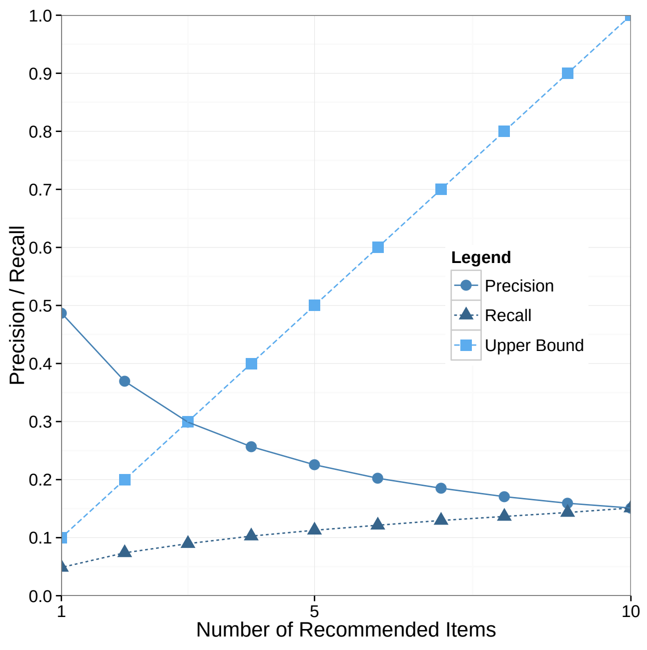


Figure 4: Precision and Recall of the Track-Based Recommender

fectly to the intended use of providing recommendations for new artists. However, this evaluation approach enabled us to find the optimal input parameters using a grid search. Secondly, as we don't have any preference values, the assumption that a certain user likes the artist he/she listened to, has to be made.

Both drawbacks can be eliminated by conducting a user-centric evaluation [5]. Thus, in a future work, it would be worth to conduct a user-experiment using the optimized recommender system.

5. RELATED WORK

As already mentioned in the introduction, there exist several other publicly available datasets suitable for music recommendations. A quick overview of these datasets is given in this Section.

One of the biggest available music datasets is the Million Song Dataset (MSD) [4]. This dataset contains information about one million songs from different sources. Beside real user play counts, it provides audio features of the songs and is therefore suitable for CF-, CB- and hybrid recommender systems. At the moment, the Taste Profile subset[8] of the MSD is bigger than the dataset presented in this work, however it was released 2011 and is therefore not as recent.

Beside the MSD, also Yahoo! published big datasets[9] containing ratings for artists and songs suitable for CF. The biggest dataset contains 136,000 songs along with ratings given by 1.8 million users. Additionally, the genre information is provided in the dataset. The data itself was gathered by monitoring users using the Yahoo! Music Services between 2002 and 2006. Again, the MSD dataset, the Yahoo dataset is less recent. Additionally to the ratings, the Yahoo dataset contains genre information which can be exploited by a hybrid recommender system.

Celma also provides a music dataset, containing data retrieved from last.fm[10], a music discovery service. It contains user, artists and play counts as well as the MusicBrainz identifiers for 360,000 users. This dataset was published in 2010 [5].

Beside the datasets presented above, which are based on data of private companies, there exist several datasets based on publicly available information. Sources exploited have been websites in general [12, 15, 14], Internet radios posting their play lists [1] and micro-blogging platforms, in particular Twitter [17, 13]. However, using these sources has a drawback: For cleaning and matching the data, high effort is necessary.

One of the most similar datasets to the dataset used in this work, is the Million Musical Tweets Dataset [11] dataset by Hauger et al. [7]. Like our dataset, it was created using the Twitter streaming API from September 2011 to April 2013, however, all tweets not containing a geolocation were removed and thus it is much smaller. The dataset contains 1,086,808 tweets by 215,375 users. Among the dataset, 25,060 unique artists have been identified [7].

Another dataset based on publicly available data which is similar to the MovieLens dataset, is the MovieTweetings dataset published by Dooms et al. [6]. The MovieTweetings dataset is continually updated and has the same format as the MovieLens dataset, in order to foster exchange. At the moment, a snapshot containing 200,000 ratings is available[12]. The dataset is generated by crawling well-structured tweets and extracting the desired information using regular expressions. Using this regular expressions, the name of the movie, the rating and the corresponding user is extracted. The data is afterwards linked to the IMDb, the Internet Movie Database [13].

6. CONCLUSION AND FUTURE WORK

In this work we have shown that the presented dataset is valuable for evaluating and benchmarking different approaches for music recommendation. We implemented a working music recommender systems, however as shown in Section 4, for a high number of recommendations the performance of our baseline recommendation approach is limited. Thus, we see a need for action at two points: First we will enrich the dataset with further context based information that is available: in this case this can be the time stamp or the geolocation. Secondly, hybrid recommender system utilizing this additional context based information are from interest. Therefore, in future works, we will focus on the implementation of such recommender systems and compare them to the presented baseline approach. First experiments were already conducted with a recommender system trying to exploit the geolocation. Two different implementations are evaluated at the moment: The first uses the normalized linear distance between two users for approximating a user

[8]`http://labrosa.ee.columbia.edu/millionsong/tasteprofile`
[9]available at: `http://webscope.sandbox.yahoo.com/catalog.php?datatype=r`
[10]`http://www.last.fm`
[11]available at: `http://www.cp.jku.at/datasets/MMTD/`
[12]`https://github.com/sidooms/MovieTweetings`
[13]`http://www.imdb.com`

similarity. The second one, which in an early stage of evaluation seems to be the more promising one, increases the user similarity if a certain distance threshold is underrun. However, there remains the open question how to determine this distance threshold.

7. REFERENCES

[1] N. Aizenberg, Y. Koren, and O. Somekh. Build your own music recommender by modeling internet radio streams. In *Proceedings of the 21st International Conference on World Wide Web (WWW 2012)*, pages 1–10. ACM, 2012.

[2] Apache Software Foundation. What is Apache Mahout?, March 2014. Retrieved July 13, 2014, from http://mahout.apache.org.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition) (ACM Press Books)*. Addison-Wesley Professional, 2 edition, 2011.

[4] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In A. Klapuri and C. Leider, editors, *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 591–596. University of Miami, 2011.

[5] Ò. Celma. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010.

[6] S. Dooms, T. De Pessemier, and L. Martens. Movietweetings: a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems at the 7th ACM Conference on Recommender Systems (RecSys 2013)*, 2013.

[7] D. Hauger, M. Schedl, A. Kosir, and M. Tkalcic. The million musical tweet dataset - what we can learn from microblogs. In A. de Souza Britto Jr., F. Gouyon, and S. Dixon, editors, *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 189–194, 2013.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, Jan. 2004.

[9] C. W. Hsu, C. C. Chang, and C. J. Lin. *A practical guide to support vector classification*. Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2003.

[10] P. Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, Feb. 1912.

[11] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. 62:81–97, 1956.

[12] A. Passant. dbrec - Music Recommendations Using DBpedia. In *Proceedings of the 9th International Semantic Web Conference (ISWC 2010)*, volume 6497 of *Lecture Notes in Computer Science*, pages 209–224. Springer Berlin Heidelberg, 2010.

[13] M. Schedl. Leveraging Microblogs for Spatiotemporal Music Information Retrieval. In *Proceedings of the 35th European Conference on Information Retrieval (ECIR 2013)*, pages 796 – 799, 2013.

[14] M. Schedl, P. Knees, and G. Widmer. Investigating web-based approaches to revealing prototypical music artists in genre taxonomies. In *Proceedings of the 1st International Conference on Digital Information Management (ICDIM 2006)*, pages 519–524. IEEE, 2006.

[15] M. Schedl, C. C. Liem, G. Peeters, and N. Orio. A Professionally Annotated and Enriched Multimodal Data Set on Popular Music. In *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys 2013)*, pages 78–83, February–March 2013.

[16] M. Schedl and D. Schnitzer. Hybrid Retrieval Approaches to Geospatial Music Recommendation. In *Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2013.

[17] E. Zangerle, W. Gassler, and G. Specht. Exploiting twitter's collective knowledge for music recommendations. In *Proceedings of the 2nd Workshop on Making Sense of Microposts (#MSM2012)*, pages 14–17, 2012.

Incremental calculation of isochrones regarding duration

Nikolaus Krismer
University of Innsbruck, Austria
nikolaus.krismer@uibk.ac.at

Günther Specht
University of Innsbruck, Austria
guenther.specht@uibk.ac.at

Johann Gamper
Free University of Bozen-Bolzano, Italy
gamper@inf.unibz.it

ABSTRACT

An isochrone in a spatial network is the minimal, possibly disconnected subgraph that covers all locations from where a query point is reachable within a given time span and by a given arrival time [5]. A novel approach for computing isochrones in multimodal spatial networks is presented in this paper. The basic idea of this incremental calculation is to reuse already computed isochrones when a new request with the same query point is sent, but with different duration. Some of the major challenges of the new calculation attempt are described and solutions to the most problematic ones are outlined on basis of the already established MINE and MINEX algorithms. The development of the incremental calculation is done by using six different cases of computation. Three of them apply to the MINEX algorithm, which uses a vertex expiration mechanism, and three cases to MINE without vertex expiration. Possible evaluations are also suggested to ensure the correctness of the incremental calculation. In the end some further tasks for future research are outlined.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Algorithms

Keywords

isochrone, incremental calculation

1. INTRODUCTION

Throughout the past years interactive online maps have become a famous tool for planning routes of any kind. Nowadays everybody with access to the internet is able to easily get support when travelling from a given point to a specific target. The websites enabling such a navigation usually calculate routes using efficient shortest path (SP) algorithms. One of the most famous examples of these tools is Google's map service named GoogleMaps[1]. For a long time it was possible to calculate routes using one transportation system (by car, by train or by bus) only. This is known as routing within unimodal spatial networks. Recent developments enabled the computation combining various transportation systems within the same route, even if some systems are bound to schedules. This has become popular under the term "multimodal routing" (or routing in multimodal spatial networks).

Less famous, but algorithmic very interesting, is to find the answer to the question where someone can travel to in a given amount of time starting at a certain time from a given place. The result is known as isochrone. Within multimodal spatial networks it has been defined by Gamper et al. [5]. Websites using isochrones include Mapnificent[2] and SimpleFleet[3] [4].

One major advantage of isochrones is that they can be used for reachability analyses of any kind. They are helpful in various fields including city planning and emergency management. While some providers, like SimpleFleet and Mapnificent, enable the computation of isochrones based on pre-calculated information or with heuristic data, the calculation of isochrones is a non-trivial and time-intense task. Although some improvements to the algorithms that can be used for isochrone computation have been published at the Free University of Bozen-Bolzano in [7], one major drawback is that the task is always performed from scratch. It is not possible to create the result of a twenty-minute-isochrone (meaning that the travelling time from/to a query point `q` is less than or equal to twenty minutes) based on the result from a 15-minute-isochrone (the travelling time is often referred to as maximal duration `dmax`). The incremental calculation could dramatically speed up the computation of isochrones, if there are other ones for the same point `q` available. This is especially true for long travel times. However, the computation based on cached results has not been realised until now and is complex. As one could see from figures 1 and 2 it is not sufficient to extend the outline of the isochrone, because there might be some network hubs (e.g. stations of the public transportation system) which extend the isochrone result into new, possibly disconnected areas.

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

[1]http://maps.google.com
[2]http://www.mapnificent.net
[3]http://www.simplefleet.eu

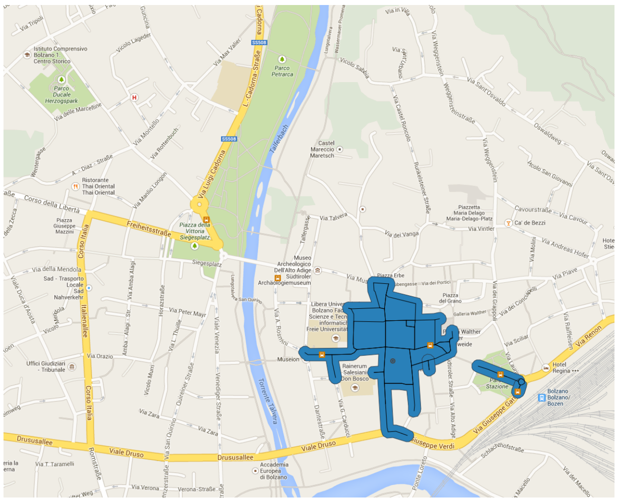


Figure 1: Isochrone with dmax of 10 minutes

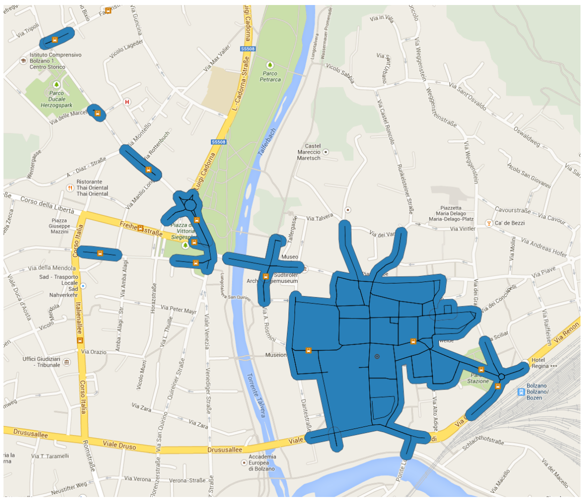


Figure 2: Isochrone with dmax of 15 minutes

This paper presents the calculation of incremental isochrones in multimodal spatial networks on top of already developed algorithms and cached results. It illustrates some ideas that need to be addressed when extending the algorithms by the incremental calculation approach. The remainder of this paper is structured as follows. Section 2 includes related work. Section 3 is split into three parts: the first part describes challenges that will have to be faced during the implementation of incremental isochrones. Possible solutions to the outlined problems are also discussed shortly here. The second part deals with different cases that are regarded during computation and how these cases differ, while the third part points out some evaluations and tests that will have to be performed to ensure the correctness of the implementation. Section 4 consists of a conclusion and lists some possible future work.

2. RELATED WORK

The calculation of isochrones in multimodal spatial networks can be done using various algorithms. The method introduced by Bauer et. al [1] suffers from high initial loading time and is limited by the available memory, since the entire network is loaded into memory at the beginning. Another algorithm, called Multimodal Incremental Network Expansion (MINE), which has been proposed by Gamper et al. [5] overcomes the limitation that the whole network has to be loaded, but is restricted to the size of the isochrone result, since all points in the isochrone are still located in memory. To overcome this limitation, the Multimodal Incremental Network Expansion with vertex eXpiration (MINEX) algorithm has been developed by Gamper et al. [6] introducing vertex expiration (also called node expiration). This mechanism eliminates unnecessary nodes from memory as soon as possible and therefore reduces the memory needed during computation.

There are some more routing algorithms that do not load the entire network into the main memory. One well-known, which is not specific for isochrone calculation, but for query processing in spatial networks in general, called Incremental Euclidean Restriction (IER), has been introduced by Papadias [8] in 2003. This algorithm loads chunks of the network into memory that are specified by the euclidean distance. The Incremental Network Expiration (INE) algorithm has also been introduced in the publication of Papadias. It is basically an extension of the Dijkstra shortest path algorithm. Deng et al. [3] improved the ideas of Papadias et al. accessing less network data to perform the calculations. The open source routing software "pgRouting"[4], which calculates routes on top of the spatial database PostGIS[5] (an extension to the well-known relational database PostgreSQL) uses an approach similar to IER. Instead of the euclidean distance it uses the network distance to load the spatial network.

In 2013 similar ideas have been applied to MINEX and resulted in an algorithm called Multimodal Range Network Expansion (MRNEX). It has been developed at the Free University of Bozen-Bolzano by Innerebner [7]. Instead of loading the needed data edge-by-edge from the network, it is loaded using chunks, like it is done in IER. Depending on their size this approach is able to reduce the number of network accesses by far and therefore reduces calculation time.

Recently the term "Optimal location queries" has been proposed by some researches like Chen et al. [2]. These queries are closely related to isochrones, since they "find a location for setting up a new server such that the maximum cost of clients being served by the servers (including the new server) is minimized".

3. INCREMENTAL CALCULATION REGARDING ISOCHRONE DURATION

In this paper the MINE and MINEX algorithms are extended by a new idea that is defined as "incremental calculation". This allows the creation of new results based on already computed and cached isochrones with different durations, but with the same query point `q` (defined as base-isochrones). This type of computation is complex, since it is not sufficient to extend an isochrone from its border points. In theory it is necessary to re-calculate the isochrone from every node in the spatial network that is part of the base-isochrone and connected to other nodes. Although this is

[4]http://pgrouting.org
[5]http://postgis.net

true for a highly connected spatial network it might not be the only or even best way for a real-world multimodal spatial network with various transportation systems. The isochrone calculation based on already known results should be doable with respect to all the isochrone's border points and all the public transportation system stations that are part of the base isochrone. These network hubs in reality are the only nodes, which can cause new, possibly disconnected areas to become part of an isochrone with different travelling time.

As it is important for the incremental calculation, the vertex expiration that is introduced by Gamper et al. in [6] will now be summarized shortly. The aim of the proposed approach is to remove loaded network nodes as soon as possible from memory. However, to keep performance high, nodes should never be double-loaded at any time and therefore they should not be eliminated from memory too soon. Removal should only occur when all computations regarding the node have been performed. States are assigned to every node to assist in finding the optimal timeslot for memory elimination. The state of a node can either be "open", "closed" or "expired". Every loaded node is labelled with the open state in the beginning. If all of its outgoing edges are traversed, its state changes to closed. However, the node itself has to be kept in memory in order to avoid cyclic network expansions. A node reaches the expired state, if all nodes in its neighbourhood reached the closed or expired state. It then can safely be removed from memory and is not available for further computations without reloading it from the network. Since this is problematic for the incremental calculation approach this aspect is described in more detail.

3.1 Challenges

There are some challenges that need to be addressed when implementing an incremental calculation for the MINE and MINEX algorithm. The most obvious problem is related to the vertex expiration of the MINEX algorithm. If nodes already expired, they will not be available to the calculation of isochrones with different durations. To take care of this problem all nodes `n` that are connected to other nodes which are not in the direct neighbourhood of `n` are added to a list `l_hubs`. These nodes are the ones we referred to as network hubs. Besides the hub node `n` itself, further information is stored in this list: the time `t` of arrival at the node and the remaining distance `d` that can be used. With this information it is possible to continue computation from any hub with a modified travelling time for the algorithms.

The list `l_hubs` needs to be stored in addition to the isochrone's maximal travelling time and the isochrone result itself, so that it can be used for incremental calculation. None of this information needs to be held in memory during computation of the base isochrone itself and is only touched on incremental calculation. Therefore, runtime and memory consumption of the isochrone algorithms will not be influenced much.

Other problems include modifications to the spatial network in combination with incremental isochrones. If there is some change applied to the underlying network, all the base isochrones can not be used for incremental calculation any more. It can not be guaranteed that the network modification does not influence the base isochrone. Changes in the schedules of one or more modalities (for example the public transportation systems) could cause problems as well, as they would also influence the base isochrone. Schedule alternations can be triggered by a service provider. Traffic jams and similar factors can lead to delays in the transportation system and thus also have to be considered. Although it should be possible to overcome both limitations or at least limit their impact, it will not be further discussed in this paper.

3.2 Types of calculation

There are six different cases that have to be kept in mind when calculating an isochrone with travelling time `dmax` using a base isochrone with duration `dmax_base`: three applying to algorithms without vertex expiration and three cases for the ones using vertex expiration.

3.2.1 *Cases* `dmax = dmax_base`

The first two and most simple cases for the MINE and MINEX algorithm, are the ones where `dmax` is equal to `dmax_base`. In these cases it is obvious that the calculation result can be returned directly without any further modification. It is not needed to respect expired nodes, since no (re)calculation needs to be performed.

3.2.2 *Cases* `dmax < dmax_base`

The third, also simple case, is the one where `dmax` is less than `dmax_base` for algorithms without vertex expiration. In this situation all nodes can be iterated and checked for suitability. If the duration is less or equal to `dmax`, then the node also belongs to the new result, otherwise it does not. In the fourth case, where the duration is less than `dmax_base` and nodes were expired (and therefore are not available in memory any more), the isochrone can be shrunk from its borders. The network hubs do not need any special treatment, since no new areas can become part of the result if the available time decreased. The only necessary task is the recalculation of the durations from the query point to the nodes in the isochrone and to possibly reload expired nodes. It either can be done from the query point or from the border points. The duration `d` from the query point `q` to a network node `n` is then equal to (assuming that the border point with the minimal distance to `n` is named `bp`):

$$d(q, n) = d(q, bp) - d(bp, n)$$

3.2.3 *Cases* `dmax > dmax_base`

The remaining two cases, where `dmax_base` is less than `dmax`, are much more complex. They differ in the fact that new, possibly disconnected areas can become part of the result and therefore it is not sufficient to look at all the base isochrones border points. The new areas become available as a result from connections caused by network hubs that often are bound to some kind of schedule. A real-world example is a train station where a train is leaving at time `t_train` due to its schedule and arriving at a remote station at or before time `dmax` (in fact any time later than `dmax_base` is feasible). The time `t_train` has to be later than the arrival time at the station (and after the isochrones starting time).

Since all network hubs are saved with all the needed information to the list `l_hubs` it is not of any interest if the algorithm uses vertex expiration or not. The points located at the isochrone's outline are still in memory. Since only network hubs can create new isochrone areas it is sufficient to grow the isochrone from its border and all the network hubs located in the isochrone. The only effect that vertex expiration causes is a smaller memory footprint of the calculation, as it would also do without incremental calculation.

In table 1 and in table 2 the recently mentioned calculation types are summarised shortly. The six different cases can be distinguished with ease using these two tables.

	MINE
`dmax` < `dmax_base`	iterating nodes from base isochrone checking if travel time is <= `dmax`
`dmax` = `dmax_base`	no change
`dmax` > `dmax_base`	extend base isochrone by border points and with list `l_hubs`

Table 1:
Incremental calculation without vertex expiration

	MINEX
`dmax` < `dmax_base`	shrink base isochrone from border
`dmax` = `dmax_base`	no change
`dmax` > `dmax_base`	extend base isochrone by border points and with list `l_hubs`

Table 2:
Incremental calculation with vertex expiration

Although the different types of computations are introduced using the MINE and MINEX algorithms they also apply to the MRNEX method. When using MRNEX the same basic idea can be used to enable incremental calculations. In addition the same advantages and disadvantages apply to the incremental calculation using MRNEX compared to MINEX that also apply to the non-incremental setup.

3.3 Evaluation

The evaluations that will need to be carried out to ensure the correctness of the implementation can be based on freely available datasets, such as OpenStreetMap[6]. Schedules from various public transportation systems could be used and since they might be subject of licensing it is planned to create some test schedules. This data can then be used as mockups and as a replacement of the license-bound real-world schedules. It is also planned to realise all the described tests in the context of a continuous integration setup. They will therefore be automatically executed ensuring the correctness throughout various software changes.

The basic idea of the evaluation is to calculate incremental isochrones on the basis of isochrones with different durations and to compare them with isochrones calculated without the incremental approach. If both results are exactly the same, the incremental calculation can be regarded as correct.

There will be various tests that need to be executed in order to cover all the different cases described in section 3.2. As such, all the cases will be performed with and without vertex expiration. The durations of the base isochrones will cover the three cases per algorithm (less than, equal to and greater than the duration of the incremental calculated isochrone). Additional tests, such as testing for vertex expiration of the incremental calculation result, will be implemented as well. Furthermore, the calculation times of both - the incremental and the non-incremental approach - will be recorded to allow comparison. The incremental calculation can only be seen as successful, if there are situations where they perform better than the common calculation. As mentioned before, this is expected to be true for at least large isochrone durations, since large portions of the spatial network does not need to be loaded then.

Besides these automatically executed tests, it will be possible to perform manual tests using a graphical user interface. This system is under heavy development at the moment and has been named IsoMap. Regardless of its young state it will enable any user to calculate isochrones with and without the incremental approach and to visually compare the results with each other.

[6]http://www.openstreetmap.org

4. CONCLUSION AND FUTURE WORK

In this paper an approach to enable the calculation of isochrones with the help of already known results was presented. The necessary steps will be realised in the near future, so that runtime comparisons between incremental calculated isochrones and such created without the presented approach will be available shortly. The ideas developed throughout this paper do not influence the time needed for calculation of base isochrones by far. The only additional complexity is generated by storing a list `l_hubs` besides the base isochrone. However, this is easy to manage and since the list does not contain any complex data structures, the changes should be doable without any noticeable consequence to the runtime of the algorithms.

Future work will extend the incremental procedure to further calculation parameters, especially to the arrival time, the travelling speed and the query point q of the isochrone. Computations on top of cached results are also realisable for changing arrival times and/or travel speeds. It should even be possible to use base isochrones with completely different query points in the context of the incremental approach. If the isochrone calculation for a duration of twenty minutes reaches a point after five minutes the 15-minute isochrone of this point has to be part of the computed result (if the arrival times are respected). Therefore, cached results can decrease the algorithm runtimes even for different query points, especially if they are calculated for points that can cause complex calculations like airports or train stations.

Open fields that could be addressed include the research of incremental calculation under conditions where public transportation system schedules may vary due to trouble in the traffic system. The influence of changes in the underlying spatial networks to the incremental procedure could also be part of future research. It is planned to use the incremental calculation approach to calculate city round trips and to allow the creation of sight seeing tours for tourists with the help of isochrones. This computation will soon be enabled in cities where it is not possible by now.

Further improvements regarding the calculation runtime of isochrones can be done as well. In this field, some examinations with different databases and even with different types of databases (in particular graph databases and other NoSQL systems) are planned.

5. REFERENCES

[1] V. Bauer, J. Gamper, R. Loperfido, S. Profanter, S. Putzer, and I. Timko. Computing isochrones in multi-modal, schedule-based transport networks. In

Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '08, pages 78:1–78:2, New York, NY, USA, 2008. ACM.

[2] Z. Chen, Y. Liu, R. C.-W. Wong, J. Xiong, G. Mai, and C. Long. Efficient algorithms for optimal location queries in road networks. In *SIGMOD Conference*, pages 123–134, 2014.

[3] K. Deng, X. Zhou, H. Shen, S. Sadiq, and X. Li. Instance optimal query processing in spatial networks. *The VLDB Journal*, 18(3):675–693, 2009.

[4] A. Efentakis, N. Grivas, G. Lamprianidis, G. Magenschab, and D. Pfoser. Isochrones, traffic and demographics. In *SIGSPATIAL/GIS*, pages 538–541, 2013.

[5] J. Gamper, M. Böhlen, W. Cometti, and M. Innerebner. Defining isochrones in multimodal spatial networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 2381–2384, New York, NY, USA, 2011. ACM.

[6] J. Gamper, M. Böhlen, and M. Innerebner. Scalable computation of isochrones with network expiration. In A. Ailamaki and S. Bowers, editors, *Scientific and Statistical Database Management*, volume 7338 of *Lecture Notes in Computer Science*, pages 526–543. Springer Berlin Heidelberg, 2012.

[7] M. Innerebner. *Isochrone in Multimodal Spatial Networks*. PhD thesis, Free University of Bozen-Bolzano, 2013.

[8] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 802–813. VLDB Endowment, 2003.

Software Design Approaches for Mastering Variability in Database Systems

David Broneske, Sebastian Dorok, Veit Köppen, Andreas Meister*
*author names are in lexicographical order
Otto-von-Guericke-University Magdeburg
Institute for Technical and Business Information Systems
Magdeburg, Germany
firstname.lastname@ovgu.de

ABSTRACT

For decades, database vendors have developed traditional database systems for different application domains with highly differing requirements. These systems are extended with additional functionalities to make them applicable for yet another data-driven domain. The database community observed that these “one size fits all” systems provide poor performance for special domains; systems that are tailored for a single domain usually perform better, have smaller memory footprint, and less energy consumption. These advantages do not only originate from different requirements, but also from differences within individual domains, such as using a certain storage device.

However, implementing specialized systems means to reimplement large parts of a database system again and again, which is neither feasible for many customers nor efficient in terms of costs and time. To overcome these limitations, we envision applying techniques known from software product lines to database systems in order to provide tailor-made and robust database systems for nearly every application scenario with reasonable effort in cost and time.

General Terms

Database, Software Engineering

Keywords

Variability, Database System, Software Product Line

1. INTRODUCTION

In recent years, data management has become increasingly important in a variety of application domains, such as automotive engineering, life sciences, and web analytics. Every application domain has its unique, different functional and non-functional requirements leading to a great diversity of *database systems* (DBSs). For example, automotive data management requires DBSs with small storage and memory consumption to deploy them on embedded devices. In contrast, big-data applications, such as in life sciences, require large-scale DBSs, which exploit newest hardware trends, e.g., vectorization and SSD storage, to efficiently process and manage petabytes of data [8]. Exploiting variability to design a tailor-made DBS for applications while making the variability manageable, that is keeping maintenance effort, time, and cost reasonable, is what we call *mastering* variability in DBSs.

Currently, DBSs are designed either as one-size-fits-all DBSs, meaning that all possible use cases or functionalities are integrated at implementation time into a single DBS, or as specialized solutions. The first approach does not scale down, for instance, to embedded devices. The second approach leads to situations, where for each new application scenario data management is reinvented to overcome resource restrictions, new requirements, and rapidly changing hardware. This usually leads to an increased time to market, high development cost, as well as high maintenance cost. Moreover, both approaches provide limited capabilities for managing variability in DBSs. For that reason, *software product line* (SPL) techniques could be applied to the data management domain. In SPLs, variants are concrete programs that satisfy the requirements of a specific application domain [7]. With this, we are able to provide tailor-made and robust DBSs for various use cases. Initial results in the context of embedded systems, expose benefits of applying SPLs to DBSs [17, 22].

The remainder of this paper is structured as follows: In Section 2, we describe variability in a database system regarding hardware and software. We review three approaches to design DBSs in Section 3, namely, the one-size-fits-all, the specialization, and the SPL approach. Moreover, we compare these approaches w.r.t. robustness and maturity of provided DBSs, the effort of managing variability, and the level of tailoring for specific application domains. Because of the superiority of the SPL approach, we argue to apply this approach to the implementation process of a DBS. Hence, we provide research questions in Section 4 that have to be answered to realize the vision of mastering variability in DBSs using SPL techniques.

2. VARIABILITY IN DATABASE SYSTEMS

Variability in a DBS can be found in software as well as hardware. Hardware variability is given due to the use of different devices with specific properties for data processing and storage. Variability in software is reflected by different functionalities that have to be provided by the DBS for a specific application. Additionally, the combination of

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

hardware and software functionality for concrete application domains increases variability.

2.1 Hardware

In the past decade, the research community exploited arising hardware features by tailor-made algorithms to achieve optimized performance. These algorithms effectively utilize, e.g., caches [19] or vector registers of *Central Processing Units* (CPUs) using AVX- [27] and SSE-instructions [28]. Furthermore, the usage of co-processors for accelerating data processing opens up another dimension [12]. In the following, we consider processing and storage devices and sketch the variability arising from their different properties.

2.1.1 Processing Devices

To sketch the heterogeneity of current systems, possible (co-)processors are summarized in Figure 1. Current systems do not only include a CPU or an *Accelerated Processing Unit* (APU), but also co-processors, such as *Many Integrated Cores* (MICs), *Graphical Processing Units* (GPUs), and *Field Programmable Gate Arrays* (FPGAs). In the following, we give a short description of varying processor properties. A more extensive overview is presented in our recent work [5].

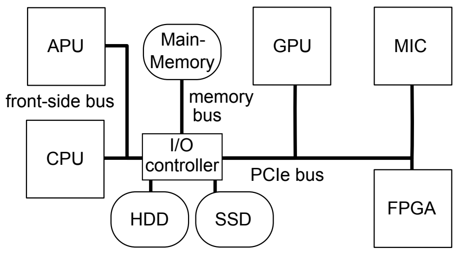


Figure 1: Future system architecture [23]

Central Processing Unit: Nowadays, CPUs consist of several independent cores, enabling parallel execution of different calculations. CPUs use pipelining, *Single Instruction Multiple Data* (SIMD) capabilities, and branch prediction to efficiently process conditional statements (e.g., if statements). Hence, CPUs are well suited for control intensive algorithms.

Graphical Processing Unit: Providing larger SIMD registers and a higher number of cores than CPUs, GPUs offer a higher degree of parallelism compared to CPUs. In order to perform calculations, data has to be transferred from main memory to GPU memory. GPUs offer an own memory hierarchy with different memory types.

Accelerated Processing Unit: APUs are introduced to combine the advantages of CPUs and GPUs by including both on one chip. Since the APU can directly access main memory, the transfer bottleneck of dedicated GPUs is eliminated. However, due to space limitations, fairly less GPU cores fit on the APU die compared to a dedicated GPU, leading to reduced computational power compared to dedicated GPUs.

Many Integrated Core: MICs use several integrated and interconnected CPU cores. With this, MICs offer a high parallelism while still featuring CPU properties. However, similar to the GPU, MICs suffer from the transfer bottleneck.

Field Programmable Gate Array: FPGAs are programmable stream processors, providing only a limited storage capacity. They consist of several independent logic cells consisting of a storage unit and a lookup table. The interconnect between logic cells and the lookup tables can be reprogrammed during run time to perform any possible function (e.g., sorting, selection).

2.1.2 Storage Devices

Similar to the processing device, current systems offer a variety of different storage devices used for data processing. In this section, we discuss different properties of current storage devices.

Hard Disk Drive: The *Hard Disk Drive* (HDD), as a non-volatile storage device, consists of several disks. The disks of an HDD rotate, while a movable head reads or writes information. Hence, sequential access patterns are well supported in contrast to random accesses.

Solid State Drive: Since no mechanical units are used, *Solid State Drives* (SSDs) support random access without high delay. For this, SSDs use flash-memory to persistently store information [20]. Each write wears out the flash cells. Consequently, the write patterns of database systems must be changed compared to HDD-based systems.

Main-Memory: While using main memory as main storage, the access gap between primary and secondary storage is removed, introducing main-memory access as the new bottleneck [19]. However, main-memory systems cannot omit secondary storage types completely, because main memory is volatile. Thus, efficient persistence mechanisms are needed for main-memory systems.

To conclude, current architectures offer several different processor and storage types. Each type has a unique architecture and specific characteristics. Hence, to ensure high performance, the processing characteristics of processors as well as the access characteristics of the underlying storage devices have to be considered. For example, if several processing devices are available within a DBS, the DBS must provide suitable algorithms and functionality to fully utilize all available devices to provide peak performance.

2.2 Software Functionality

Besides hardware, DBS functionality is another source of variability in a DBS. In Figure 2, we show an excerpt of DBS functionalities and their dependencies. For example, for different application domains different query types might be interesting. However, to improve performance or development cost, only required query types should be used within a system. This example can be extended to other functional requirements. Furthermore, a DBS provides database operators, such as aggregation functions or joins. Thereby, database operators perform differently depending on the used storage and processing model [1]. For example, row stores are very efficient when complete tuples should be retrieved, while column stores in combination with operator-at-a-time processing enable fast processing of single columns [18]. Another technique to enable efficient access to data is to use index structures. Thereby, the choice of an appropriate index structure for the specific data and query types is essential to guarantee best performance [15, 24].

Note, we omit comprehensive relationships between functionalities properties in Figure 2 due to complexity. Some

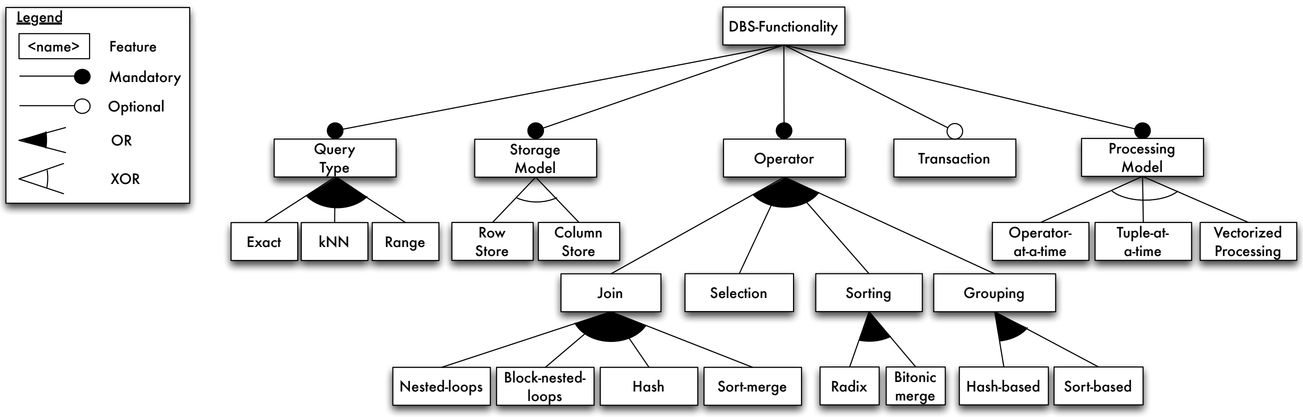


Figure 2: Excerpt of DBMS-Functionality

functionalities are mandatory in a DBS and others are optional, such as support for transactions. Furthermore, it is possible that some alternatives can be implemented together and others only exclusively.

2.3 Putting it all together

So far, we considered variability in hardware and software functionality separately. When using a DBS for a specific application domain, we also have to consider special requirements of this domain as well as the interaction between hardware and software.

Special requirements comprise functional as well as non-functional ones. Examples for functional requirements are user-defined aggregation functions (e.g., to perform genome analysis tasks directly in a DBS [9]). Other applications require support for spatial queries, such as geo-information systems. Thus, special data types as well as index structures are required to support these queries efficiently.

Besides performance, memory footprint and energy efficiency are other examples for non-functional requirements. For example, a DBS for embedded devices must have a small memory footprint due to resource restrictions. For that reason, unnecessary functionality is removed and data processing is implemented as memory efficient as possible. In this scenario, tuple-at-a-time processing is preferred, because intermediate results during data processing are smaller than in operator-at-a-time processing, which leads to less memory consumption [29].

In contrast, in large-scale data processing, operators should perform as fast as possible by exploiting underlying hardware and available indexes. Thereby, exploiting underlying hardware is another source of variability as different processing devices have different characteristics regarding processing model and data access [6]. To illustrate this fact, we depict different storage models for DBS in Figure 2. For example, column-storage is preferred on GPUs, because row-storage leads to an inefficient memory access pattern that deteriorates the possible performance benefits of GPUs [13].

3. APPROACHES TO DESIGN TAILOR-MADE DATABASE SYSTEMS

The variability in hardware and software of DBSs can be exploited to tailor database systems for nearly every database-application scenario. For example, a DBS for high-performance analysis can exploit newest hardware features, such as SIMD, to speed up analysis workloads. Moreover, we can meet limited space requirements in embedded systems by removing unnecessary functionality [22], such as the support for range queries. However, exploiting variability is one part of mastering variability in DBSs. The second part is to manage variability efficiently to reduce development and maintenance effort.

In this section, first, we describe three different approaches to design and implement DBSs. Then, we compare these approaches regarding their applicability to arbitrary database scenarios. Moreover, we assess the effort to manage variability in DBSs. Besides managing and exploiting the variability in database systems, we also consider the robustness and correctness of tailor-made DBSs created by using the discussed approaches.

3.1 One-Size-Fits-All Design Approach

One way to design database systems is to integrate all conceivable data management functionality into one single DBS. We call this approach the *one-size-fits-all* design approach and DBSs designed according to this approach *one-size-fits-all* DBSs. Thereby, support for hardware features as well as DBMS functionality are integrated into one code base. Thus, one-size-fits-all DBSs provide a rich set of functionality. Examples of database systems that follow the one-size-fits-all approach are PostgreSQL, Oracle, and IBM DB2. As one-size-fits-all DBSs are monolithic software systems, implemented functionality is highly interconnected on the code level. Thus, removing functionality is mostly not possible.

DBSs that follow the one-size-fits-all design approach aim at providing a comprehensive set of DBS functionality to deal with most database application scenarios. The claim for generality often introduces functional overhead that leads to performance losses. Moreover, customers pay for functionality they do not really need.

3.2 Specialization Design Approach

In contrast to one-size-fits-all DBSs, DBSs can also be designed and developed to fit very specific use cases. We call this design approach the *specialization design approach* and DBSs designed accordingly, *specialized* DBSs. Such DBSs are designed to provide only that functionality that is needed for the respective use case, such as text processing, data warehousing, or scientific database applications [25]. Specialized DBSs are often completely redesigned from scratch to meet application requirements and do not follow common design considerations for database systems, such as locking and latching to guarantee multi-user access [25]. Specialized DBSs remove the overhead of unneeded functionality. Thus, developers can highly focus on exploiting hardware and functional variability to provide tailor-made DBSs that meet high-performance criteria or limited storage space requirements. Therefore, huge parts of the DBS (if not all) must be newly developed, implemented, and tested which leads to duplicate implementation efforts, and thus, increased development costs.

3.3 Software Product Line Design Approach

In the specialization design approach, a new DBS must be developed and implemented from scratch for every conceivable database application. To avoid this overhead, the *SPL design approach* reuses already implemented and tested parts of a DBS to create a tailor-made DBS.

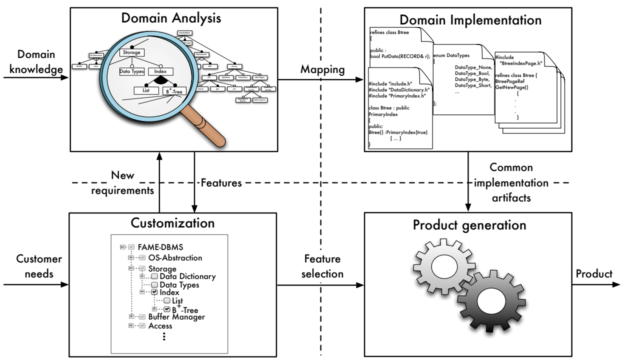


Figure 3: Managing Variability

To make use of SPL techniques, a special workflow has to be followed which is sketched in Figure 3 [2]. At first, the domain is modeled, e.g., by using a feature model – a tree-like structure representing features and their dependencies. With this, the variability is captured and implementation artifacts can be derived for each feature. The second step, the domain implementation, is to implement each feature using a compositional or annotative approach. The third step of the workflow is to customize the product – in our case, the database system – which will be generated afterwards.

By using the SPL design approach, we are able to implement a database system from a set of features which are mostly already provided. In best case, only non-existing features must be implemented. Thus, the feature pool constantly grows and features can be reused in other database systems. Applying this design approach to DBSs enables to create DBSs tailored for specific use cases while reducing functional overhead as well as development time. Thus, the *SPL design approach* aims at the middle ground of the one-size-fits-all and the specialization design approach.

3.4 Characterization of Design Approaches

In this section, we characterize the three design approaches discussed above regarding:

a) general *applicability* to arbitrary database applications,
b) effort for *managing* variability, and
c) *maturity* of the deployed database system.

Although the one-size-fits-all design approach aims at providing a comprehensive set of DBS functionality to deal with most database application scenarios, a one-size-fits-all database is not applicable to use cases in automotive, embedded, and ubiquitous computing. As soon as tailor-made software is required to meet especially storage limitations, one-size-fits-all database systems cannot be used. Moreover, specialized database systems for one specific use case outperform one-size-fits-all database systems by orders of magnitude [25]. Thus, although one-size-fits-all database systems can be applied, they are often not the best choice regarding performance. For that reason, we consider the applicability of one-size-fits-all database systems to arbitrary use cases as limited. In contrast, specialized database systems have a very good applicability as they are designed for that purpose.

The applicability of the SPL design approach is good as it also creates database systems tailor-made for specific use cases. Moreover, the SPL design approach explicitly considers variability during software design and implementation and provides methods and techniques to manage it [2]. For that reason, we assess the effort of managing variability with the SPL design approach as lower than managing variability using a one-size-fits-all or specialized design approach.

We assess the maturity of one-size-fits-all database systems as very good, as these systems are developed and tested over decades. Specialized database systems are mostly implemented from scratch, so, the possibility of errors in the code is rather high, leading to a moderate maturity and robustness of the software. The SPL design approach also enables the creation of tailor-made database systems, but from approved features that are already implemented and tested. Thus, we assess the maturity of database systems created via the SPL design approach as good.

In Table 1, we summarize our assessment of the three software design approaches regarding the above criteria.

Criteria	Approach		
	One-Size-Fits-All	Specialization	SPL
a) Applicability	−	++	+
b) Management effort	−	−	+
c) Maturity	++	○	+

Table 1: Characteristics of approaches
Legend: ++ = very good, + = good, ○ = moderate, − = limited

The one-size-fits-all and the specialization design approach are each very good in one of the three categories respectively. The one-size-fits-all design approach provides robust and mature DBSs. The specialization design approach provides greatest applicability and can be used for nearly every use case. Whereas the SPL design approach provides a balanced assessment regarding all criteria. Thus, against the backdrop of increasing variability due to increasing variety of use cases and hardware while guaranteeing mature and robust DBSs, SPL design approach should be applied to develop future DBSs. Otherwise, development costs for yet another DBS which has to meet special requirements of the next data-driven domain will limit the use of DBSs in such fields.

4. ARISING RESEARCH QUESTIONS

Our assessment in the previous section shows that the SPL design approach is the best choice for mastering variability in DBSs. To the best of our knowledge, the SPL design approach is applied to DBSs only in academic settings (e.g., in [22]).Hereby, the previous research were based on BerkeleyDB. Although BerkeleyDB offers the essential functionality of DBSs (e.g., a processing engine), several functionality of relational DBSs were missing (e.g., optimizer, SQL-interface). Although these missing functionality were partially researched (e.g., storage manager [16] and the SQL parser [26]), no holistic evaluation of a DBS SPL is available. Especially, the optimizer in a DBS (e.g., query optimizer) with a huge number of crosscutting concerns is currently not considered in research. So, there is still the need for research to fully apply SPL techniques to all parts of a DBS. Specifically, we need methods for modeling variability in DBSs and efficient implementation techniques and methods for implementing variability-aware database operations.

4.1 Modeling

For modeling variability in feature-oriented SPLs, feature models are state of the art [4]. A feature model is a set of features whose dependencies are hierarchically modeled. Since variability in DBSs comprises hardware, software, and their interaction, the following research questions arise:

RQ-M1: What is a good granularity for modeling a variable DBS?
In order to define an SPL for DBSs, we have to model features of a DBS. Such features can be modeled with different levels of granularity [14]. Thus, we have to find an applicable level of granularity for modeling our SPL for DBSs. Moreover, we also have to consider the dependencies between hardware and software. Furthermore, we have to find a way to model the hardware and these dependencies. In this context, another research questions emerges:

RQ-M2: What is the best way to model hardware and its properties in an SPL?
Hardware has become very complex and researchers demand to develop a better understanding of the impact of hardware on the algorithm performance, especially when parallelized [3, 5]. Thus, the question arises what properties of the hardware are worth to be captured in a feature model.

Furthermore, when thinking about numerical properties, such as CPU frequency or amount of memory, we have to find a suitable technique to represent them in feature models. One possibility are *attributes* of extended feature-models [4], which have to be explored for applicability.

4.2 Implementing

In the literature, there are several methods for implementing an SPL. However, most of them are not applicable to our use case. Databases rely on highly tuned operations to achieve peak performance. Thus, variability-enabled implementation techniques must not harm the performance, which leads to the research question:

RQ-I1: What is a good variability-aware implementation technique for an SPL of DBSs?
Many state of the art implementation techniques are based on inheritance or additional function calls, which causes performance penalties. A technique that allows for variability without performance penalties are preprocessor directives. However, maintaining preprocessor-based SPLs is horrible, which accounts this approach the name *#ifdef Hell* [11, 10]. So, there is a trade-off between performance and maintainability [22], but also granularity [14]. It could be beneficial for some parts of DBS to prioritize maintainability and for others performance or maintainability.

RQ-I2: How to combine different implementation techniques for SPLs?
If the answer of RQ-I1 is to use different implementation techniques within the same SPL, we have to find an approach to combine these. For example, database operators and their different hardware optimization must be implemented using annotative approaches for performance reasons, but the query optimizer can be implemented using compositional approaches supporting maintainability; the SPL product generator has to be aware of these different implementation techniques and their interactions.

RQ-I3: How to deal with functionality extensions?
Thinking about changing requirements during the usage of the DBS, we should be able to extend the functionality in the case user requirements change. Therefore, we have to find a solution to deploy updates from an extended SPL in order to integrate the new requested functionality into a running DBS. Some ideas are presented in [21], however, due to the increased complexity of hardware and software requirements an adaption or extension is necessary.

4.3 Customization

In the final customization, features of the product line are selected that apply to the current use case. State of the art approaches just list available features and show which features are still available for further configuration. However, in our scenario, it could be helpful to get further information of the configuration possibilities. Thus, another research question is:

RQ-C1: How to support the user to obtain the best selection?
In fact, it is possible to help the user in identifying suitable configurations for his use case. If he starts to select functionality that has to be provided by the generated system, we can give him advice which hardware yields the best performance for his algorithms. However, to achieve this we have to investigate another research question:

RQ-C2: How to find the optimal algorithms for a given hardware?
To answer this research question, we have to investigate the relation between algorithmic design and the impact of the hardware on the execution. Hence, suitable properties of algorithms have to be identified that influence performance on the given hardware, e.g., access pattern, size of used data structures, or result sizes.

5. CONCLUSIONS

DBSs are used for more and more use cases. However, with an increasing diversity of use cases and increasing heterogeneity of available hardware, it is getting more challenging to design an optimal DBS while guaranteeing low implementation and maintenance effort at the same time. To solve this issue, we review three design approaches, namely the one-size-fits-all, the specialization, and the software product line design approach. By comparing these three design approaches, we conclude that the SPL design approach is a promising way to master variability in DBSs and to provide mature data management solutions with reduced implementation and maintenance effort. However, there is currently no comprehensive software product line in the field of DBSs available. Thus, we present several research questions that have to be answered to fully apply the SPL design approach on DBSs.

6. ACKNOWLEDGMENTS

This work has been partly funded by the German BMBF under Contract No. 13N10818 and Bayer Pharma AG.

7. REFERENCES

[1] D. J. Abadi, S. R. Madden, and N. Hachem. Column-stores vs. Row-stores: How Different Are They Really? In *SIGMOD*, pages 967–980. ACM, 2008.

[2] S. Apel, D. Batory, C. Kästner, and G. Saake. *Feature-Oriented Software Product Lines*. Springer, 2013.

[3] C. Balkesen, G. Alonso, J. Teubner, and M. T. Özsu. Multi-Core, Main-Memory Joins: Sort vs. Hash Revisited. *PVLDB*, 7(1):85–96, 2013.

[4] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated Analysis of Feature Models 20 Years Later: A Literature Review. *Inf. Sys.*, 35(6):615–636, 2010.

[5] D. Broneske, S. Breß, M. Heimel, and G. Saake. Toward Hardware-Sensitive Database Operations. In *EDBT*, pages 229–234, 2014.

[6] D. Broneske, S. Breß, and G. Saake. Database Scan Variants on Modern CPUs: A Performance Study. In *IMDM@VLDB*, 2014.

[7] K. Czarnecki and U. W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. ACM Press/Addison-Wesley Publishing Co., 2000.

[8] S. Dorok, S. Breß, H. Läpple, and G. Saake. Toward Efficient and Reliable Genome Analysis Using Main-Memory Database Systems. In *SSDBM*, pages 34:1–34:4. ACM, 2014.

[9] S. Dorok, S. Breß, and G. Saake. Toward Efficient Variant Calling Inside Main-Memory Database Systems. In *BIOKDD-DEXA*. IEEE, 2014.

[10] J. Feigenspan, C. Kästner, S. Apel, J. Liebig, M. Schulze, R. Dachselt, M. Papendieck, T. Leich, and G. Saake. Do Background Colors Improve Program Comprehension in the #ifdef Hell? *Empir. Softw. Eng.*, 18(4):699–745, 2013.

[11] J. Feigenspan, M. Schulze, M. Papendieck, C. Kästner, R. Dachselt, V. Köppen, M. Frisch, and G. Saake. Supporting Program Comprehension in Large Preprocessor-Based Software Product Lines. *IET Softw.*, 6(6):488–501, 2012.

[12] B. He, M. Lu, K. Yang, R. Fang, N. K. Govindaraju, Q. Luo, and P. V. Sander. Relational Query Coprocessing on Graphics Processors. *TODS*, 34(4):21:1–21:39, 2009.

[13] B. He and J. X. Yu. High-throughput Transaction Executions on Graphics Processors. *PVLDB*, 4(5):314–325, Feb. 2011.

[14] C. Kästner, S. Apel, and M. Kuhlemann. Granularity in Software Product Lines. In *ICSE*, pages 311–320. ACM, 2008.

[15] V. Köppen, M. Schäler, and R. Schröter. Toward Variability Management to Tailor High Dimensional Index Implementations. In *RCIS*, pages 452–457. IEEE, 2014.

[16] T. Leich, S. Apel, and G. Saake. Using Step-wise Refinement to Build a Flexible Lightweight Storage Manager. In *ADBIS*, pages 324–337. Springer-Verlag, 2005.

[17] J. Liebig, S. Apel, C. Lengauer, and T. Leich. RobbyDBMS: A Case Study on Hardware/Software Product Line Engineering. In *FOSD*, pages 63–68. ACM, 2009.

[18] A. Lübcke, V. Köppen, and G. Saake. Heuristics-based Workload Analysis for Relational DBMSs. In *UNISCON*, pages 25–36. Springer, 2012.

[19] S. Manegold, P. A. Boncz, and M. L. Kersten. Optimizing Database Architecture for the New Bottleneck: Memory Access. *VLDB J.*, 9(3):231–246, 2000.

[20] R. Micheloni, A. Marelli, and K. Eshghi. *Inside Solid State Drives (SSDs)*. Springer, 2012.

[21] M. Rosenmüller. *Towards Flexible Feature Composition: Static and Dynamic Binding in Software Product Lines*. Dissertation, University of Magdeburg, Germany, June 2011.

[22] M. Rosenmüller, N. Siegmund, H. Schirmeier, J. Sincero, S. Apel, T. Leich, O. Spinczyk, and G. Saake. FAME-DBMS: Tailor-made Data Management Solutions for Embedded Systems. In *SETMDM*, pages 1–6. ACM, 2008.

[23] M. Saecker and V. Markl. Big Data Analytics on Modern Hardware Architectures: A Technology Survey. In *eBISS*, pages 125–149. Springer, 2012.

[24] M. Schäler, A. Grebhahn, R. Schröter, S. Schulze, V. Köppen, and G. Saake. QuEval: Beyond High-Dimensional Indexing à la Carte. *PVLDB*, 6(14):1654–1665, 2013.

[25] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The End of an Architectural Era (It's Time for a Complete Rewrite). In *VLDB*, pages 1150–1160, 2007.

[26] S. Sunkle, M. Kuhlemann, N. Siegmund, M. Rosenmüller, and G. Saake. Generating Highly Customizable SQL Parsers. In *SETMDM*, pages 29–33. ACM, 2008.

[27] T. Willhalm, I. Oukid, I. Müller, and F. Faerber. Vectorizing Database Column Scans with Complex Predicates. In *ADMS@VLDB*, pages 1–12, 2013.

[28] J. Zhou and K. A. Ross. Implementing Database Operations Using SIMD Instructions. In *SIGMOD*, pages 145–156. ACM, 2002.

[29] M. Zukowski. *Balancing Vectorized Query Execution with Bandwidth-Optimized Storage*. PhD thesis, CWI Amsterdam, 2009.

PageBeat - Zeitreihenanalyse und Datenbanken

Andreas Finger
Institut für Informatik
Universität Rostock
18051 Rostock
andreas.finger@uni-rostock.de

Ilvio Bruder
Institut für Informatik
Universität Rostock
18051 Rostock
ilvio.bruder@uni-rostock.de

Andreas Heuer
Institut für Informatik
Universität Rostock
18051 Rostock
andreas.heuer@uni-rostock.de

Steffen Konerow
Mandarin Medien GmbH
Graf-Schack-Allee 9
19053 Schwerin
sk@mandarin-medien.de

Martin Klemkow
Mandarin Medien GmbH
Graf-Schack-Allee 9
19053 Schwerin
mk@mandarin-medien.de

ABSTRACT

Zeitreihendaten und deren Analyse sind in vielen Anwendungsbereichen eine wichtiges Mittel zur Bewertung, Steuerung und Vorhersage. Für die Zeitreihenanalyse gibt es eine Vielzahl von Methoden und Techniken, die in Statistiksoftware umgesetzt und heutzutage komfortabel auch ohne eigenen Implementierungsaufwand einsetzbar sind. In den meisten Fällen hat man es mit massenhaft Daten oder auch Datenströmen zu tun. Entsprechend gibt es spezialisierte Management-Tools, wie Data Stream Management Systems für die Verarbeitung von Datenströmen oder Time Series Databases zur Speicherung und Anfrage von Zeitreihen. Der folgende Artikel soll hier zu einen kleinen Überblick geben und insbesondere die Anwendbarkeit an einem Projekt zur Analyse und Vorhersage von Zuständen von Webservern veranschaulichen. Die Herausforderung innerhalb dieses Projekts „PageBeat“ ist es massenhaft Zeitreihen in Echtzeit zu analysieren und für weiterführende Analyseprozesse zu speichern. Außerdem sollen die Ergebnisse zielgruppenspezifisch aufbereitet und visualisiert sowie Benachrichtigungen ausgelöst werden. Der Artikel beschreibt den im Projekt gewählten Ansatz und die dafür eingesetzten Techniken und Werkzeuge.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Big Data, Data Mining and Knowledge Discovery, Streaming Data

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

Keywords

Datenanalyse, R, Time Series Database

1. EINFÜHRUNG

Zeitreihen sind natürlich geordnete Folgen von Beobachtungswerten. Die Zeitreihenanalyse beschäftigt sich mit Methoden zur Beschreibung dieser Daten etwa mit dem Ziel der Analyse (Verstehen), Vorhersage oder Kontrolle (Steuerung) der Daten. Entsprechende Methoden stehen in freier und kommerzieller Statistiksoftware wie R[1] , Matlab[2], Weka[3] [7], SPSS[4] und anderen zur Verfügung wodurch eine komfortable Datenauswertung ohne eigenen Implementierungsaufwand ermöglicht wird. Verfahren zur Zeitreihenanalyse sind etwa die Ermittlung von Trends und Saisonalität, wobei der Trend den längerfristigen Anstieg und die Saisonalität wiederkehrende Muster (jedes Jahr zu Weihnachten steigen die Verkäufe) repräsentieren. So werden Abhängigkeiten in den Daten untersucht, welche eine Prognose zukünftiger Werte mit Hilfe geeigneter Modelle ermöglichen. In einer Anwendung die in hoher zeitlicher Auflösung eine Vielzahl von Messwerten erfasst, entstehen schnell große Datenmengen. Diese sollen in Echtzeit analysiert werden und gegebenenfalls zur weiteren Auswertung persistent gespeichert werden. Hierfür existieren zum Einen Ansätze aus der Stromdatenverarbeitung und zum Anderen zur Speicherung von auf Zeitreihen spezialisierte Datenbanksysteme (Time Series Databases). Da statistische Analysen etwa mit standalone R Anwendungen nur funktionieren, solange die zu analysierenden Daten die Größe des Hauptspeichers nicht überschreiten, ist es notwendig die statistische Analyse in Daten-

[1]R – Programmiersprache für statistische Rechnen und Visualisieren von der R Foundation for Statistical Computing, `http://www.r-project.org`.

[2]Matlab – kommerzielle Software zum Lösen Veranschaulichen mathematischer Probleme vom Entwickler The Mathworks, `http://www.mathworks.de`.

[3]Weka – Waikato Environment for Knowledge Analysis, ein Werkzeugkasten für Data Mining und Maschinelles Lernen von der University of Waikato, `http://www.cs.waikato.ac.nz/ml/weka/`.

[4]SPSS – kommerzielle Statistik- und Analysesoftware von IBM, `http://www-01.ibm.com/software/de/analytics/spss`.

banksysteme zu integrieren. Ziel ist dabei der transparente Zugriff auf partitionierte Daten und deren Analyse mittels partitionierter statistischen Modelle. In [6] werden verschiedene Möglichkeiten der Integration beschrieben und sind in Prototypen basierend auf postgreSQL bereits umgesetzt. Auch kommerzielle Produkte wie etwa Oracle R Enterprise[4] integrieren statistische Analyse auf Datenbankebene. Im Open-Source-Bereich existiert eine Vielzahl von Ansätzen zum Umgang mit Zeitreihen, wobei uns InfluxDB[5] als besonders geeignetes Werkzeug aufgefallen ist.
Die Herausforderung innerhalb des im Weiteren beschriebenen Projekts „PageBeat“ ist es innovative und anwendungsreife Open-Source-Lösungen aus den genannten Bereichen zur Verarbeitung großer Zeitreihendaten innerhalb des Projektes miteinander zu kombinieren. Im Folgenden wird das Projekt vorgestellt, um dann verschiedene in Frage kommenden Techniken und abschließend das gewählte Konzept und erste Ergebnisse vorzustellen.

2. PROJEKT PAGEBEAT

Mit „PageBeat“ wird eine als „Software as a Service“ (SAAS) angebotene Softwaresuite speziell zur Beobachtung und Überprüfung von Webanwendungen entwickelt. Dies erfolgt zunächst im Rahmen eines vom Bundeswirtschaftsministerium geförderten ZIM-Kooperationsprojektes. Ziel der Software ist das Beobachten des und das Berichten über den aktuellen technischen Status einer Webanwendung (Website, Content Management System, E-Commerce System, Webservice) sowie das Prognostizieren technischer Probleme anhand geeigneter Indikatoren (Hardware- und Softwarespezifische Parameter). Die Berichte werden dabei für unterschiedliche Nutzergruppen (Systemadministratoren, Softwareentwickler, Abteilungsleiter, Geschäftsführung, Marketing) und deren Anforderungen aufbereitet und präsentiert. Mittels „PageBeat“ werden somit automatisiert Fehlerberichte erstellt, die über akute sowie vorhersehbare kritische Änderungen der Betriebsparameter einer Webanwendung informieren und zielgruppenspezifisch dargestellt werden.
Bei den zugrunde liegenden Kennzahlen handelt es sich um eine Reihe von Daten, die den Zustand des Gesamtsystems im Anwendungsbereich Webshopsysteme widerspiegeln. Dies sind Kennzahlen des Serverbetriebssystems (etwa CPU oder RAM Auslastung) als auch anwendungsspezifische Kenndaten (etwa die Laufzeit von Datenbankanfragen). Diese Daten sind semantisch beschrieben und entsprechende Metadaten sind in einer Wissensbasis abgelegt. Darüber hinaus ist die Verwendung weiterer Kontextinformationen angedacht, die Einfluss auf den technischen Status des Systems haben können. Hierbei kann es sich etwa um Wetterdaten handeln: beim Kinobetreiber Cinestar ist ein regnerisches Wochenende vorausgesagt, dass auf eine hohe Auslastung des Kinokartenonlineshops schließen lässt. Ein anderes Beispiel wären Informationen aus der Softwareentwicklung: bei Codeänderungen mit einem bestimmten Zeitstempel können Effekte in den Auswertungen zu diesem Zeitpunkt nachgewiesen werden. Das Ändern oder Hinzufügen bzw. Beachten von relevanten Inhalten auf den Webseiten können signifikante Änderungen in Analysen ergeben, z.B. bei der Schaltung von Werbung oder bei Filmbewertungen zu neu anlaufenden Filmen auf sozialen Plattformen.

[5]InfluxDB - An open-source distributed time series database with no external dependencies. `http://influxdb.com`.

Es wird derzeit ein möglichst breites Spektrum an Daten in hoher zeitlicher Auflösung erfasst, um in einem Prozess der Datenexploration auf Zusammenhänge schließen zu können, die zunächst nicht offensichtlich sind bzw. um Vermutungen zu validieren. Derzeit werden über 300 Kennzahlen alle 10 s auf 14 Servern aus 9 Kundenprojekten abgetastet. Diese Daten werden gespeichert und außerdem unmittelbar weiterverarbeitet. So findet etwa ein Downsampling für alle genannten 300 Kennzahlen statt. Dabei werden die zeitliche Auflösung unter Verwendung verschiedener Aggregatfunktionen auf Zeitfenster unterschiedlicher Größe reduziert und die Ergebnisse gespeichert. Andere Analysefunktionen quantisieren die Werte hinsichtlich ihrer Zugehörigkeit zu Statusklassen (etwa optimal, normal, kritisch) und speichern die Ergebnisse ebenfalls. So entstehen sehr schnell große Datenmengen. Derzeit enthält der Datenspeicher etwa 40 GB Daten und wir beobachten bei der aktuellen Anzahl beobachteter Werte einen Zuwachs von etwa 1 GB Daten pro Woche. Auf Basis der erhobenen Daten müssen zeitkritische Analysen wie etwa eine Ausreißererkennung oder die Erkennung kritischer Muster nahezu in Echtzeit erfolgen, um Kunden ein rechtzeitiges Eingreifen zu ermöglichen. Weiterhin soll eine Vorhersage zukünftiger Werte frühzeitig kritische Entwicklungen aufzeigen. Die Herausforderung im Projekt ist die Bewältigung des großen Datenvolumens unter Gewährleistung einer echtzeitnahen Bearbeitung durch Analysefunktionen.

3. ZEITREIHENANALYSE UND DATENBANKEN

Im Rahmen der Evaluierung von für das Projekt geeigneter Software haben wir verschiedene Ansätze zur Datenstromverarbeitung und der Analyse und Verwaltung von Zeitreihen untersucht. Ziel war die Verwendung frei verfügbarer Software die zudem auf im Unternehmen vorhandener technischer Expertise basiert.

3.1 Data Stream Management Systems

Die Verarbeitung kontinuierlicher Datenströme stellt einen Aspekt unseres Projektes dar. Datenstromverarbeitende Systeme bieten hierzu die Möglichkeit kontinuierliche Anfragen auf in temporäre Relationen umgewandelte Datenströme zu formulieren. Dies kann etwa mit Operatoren der im Projekt Stream[1] entwickelten an SQL angelehnten Continuous Query Language[2] erfolgen. Sollen nun komplexere Muster in Datenströmen erkannt werden, spricht man auch von der Verarbeitung komplexer Ereignisse. Im Kontext unseres Projektes entspricht so ein Muster etwa dem Anstieg der Aufrufe einer Seite aufgrund einer Marketingaktion, welcher eine höhere Systemauslastung zur Folge hat (cpu-usage), was sich wiederum in steigenden time-to-first-byte-Werten niederschlägt und in einem kritischen Bereich zur Benachrichtigung oder gar zur automatischen Aufstockung der verfügbaren Ressourcen führen soll. Complex Event Processing Systems wie Esper[5] bieten die Möglichkeit Anfragen nach solchen Mustern auf Datenströme zu formulieren und entsprechende Reaktionen zu implementieren. Da etwa Esper als eines der wenigen frei verfügbaren und für den produktiven Einsatz geeigneten Systeme, in Java und .net implementiert ist, entsprechende Entwicklungskapazitäten jedoch nicht im Unternehmen zur Verfügung stehen, wird im Projekt keines der erwähnten DSMS oder CEPS zum Einsatz kommen. Deren Architektur diente jedoch zur Orientierung bei der Entwicklung eines eigenen mit im Unternehmen eingesetzten Techniken (etwa node.js[6], RabbitMQ[7], MongoDB[8], u.a.) Systems für PageBeat.

3.2 Werkzeuge zur Datenanalyse

Zur statistischen Auswertung der Daten im Projekt werden Werkzeuge benötigt, die es ohne großen Implementierungsaufwand ermöglichen verschiedene Verfahren auf die erhobenen Daten anzuwenden und auf ihre Eignung hin zu untersuchen. Hierfür stehen verschiedene mathematische Werkzeuge zur Verfügung. Kommerzielle Produkte sind etwa die bereits erwähnten Matlab oder SPSS. Im Bereich frei verfügbarer Software kann man auf WEKA und vor allem R zurückgreifen. Besonders R ist sehr weit verbreitet und wird von einer großen Entwicklergemeinde getragen. Dadurch sind für R bereits eine Vielzahl von Verfahren zur Datenaufbereitung und deren statistischer Analyse bis hin zur entsprechenden Visualisierung implementiert. Gerade in Bezug auf die Analyse von Zeitreihen ist R aufgrund vielfältiger verfügbarer Pakete zur Zeitreihenanalyse gegenüber WEKA die geeignetere Wahl. Mit RStudio[9] steht außerdem eine komfortable Entwicklungsumgebung zur Verfügung. Weiterhin können mit dem Web Framework Shiny[10] schnell R Anwendungen im Web bereit gestellt werden und unterstützt somit eine zügige Anwendungsentwicklung. Somit stellt R mit den zugehörigen Erweiterungen die für das Projekt geeignete Umgebung zur Evaluierung von Datenanalyseverfahren und zur Datenexploration dar. Im weiteren Verlauf des Projektes und in der Überführung in ein produktives System wird die Datenanalyse, etwa die Berechnung von Vorhersagen, innerhalb von node.js reimplementiert.

3.3 Datenbankunterstützung

Klassische objektrelationale DBMS wie Oracle[11], IBM Informix[12] oder PostgreSQL[13] unterstützen in unterschiedlichem Umfang die Speicherung, Anfrage und Auswertung von Zeitreihen. PostgreSQL ermöglicht bswp. die Verwendung von Fensterfunktionen etwa zur Berechnung von Aggregatwerten für entsprechende Zeitabschnitte. Die IBM Informix TimeSeries Solution[3] stellt Container zur Speicherung von Zeitreihendaten zur Verfügung, wodurch der Speicherplatzbedarf optimiert, die Anfragegeschwindigkeit erhöht sowie die Komplexität der Anfragen reduziert werden sollen. Oracle unterstützt nicht nur die Speicherung und Anfrage von Zeitreihen, sondern integriert darüber hinaus umfassende statistische Analysefunktionalität mittels Oracle R Technologies[4]. Dabei hat der R-Anwendungsentwickler die Möglichkeit Oracle Data Frames zu verwenden, um Datenlokalität zu erreichen. Dabei wird der Code in der Oracle-Umgebung ausgeführt, dort wo die Daten liegen und nicht umgekehrt. Außerdem erfolgt so ein transparenter Zugriff auf die Daten und Aspekte der Skalierung werden durch das DBMS abgewickelt.
Neben den klassischen ORDBMS existieren eine Vielzahl von auf Zeitserien spezialisierte Datenbanken wie OpenTSDB[14], KairosDB[15], RRDB[16]. Dabei handelt es sich jeweils um einen auf Schreibzugriffe optimierten Datenspeicher in Form einer schemalosen Datenbank und darauf zugreifende Anfrage-, Analyse- und Visualisierungsfunktionalität. Man sollte sie deshalb vielmehr als Ereignis-Verarbeitungs- oder Monitoring-Systeme bezeichnen. Neben den bisher genannten Zeitseriendatenbanken ist uns bei der Recherche von für das Projekt geeigneter Software InfluxDB[17] aufgefallen. InfluxDB verwendet Googles auf Log-structured merge-trees basierenden key-value Store LevelDB[18] und setzt somit auf eine hohen Durchsatz bzgl. Schreiboperationen. Einen Nachteil hingegen stellen langwierige Löschoperationen ganzer nicht mehr benötigter Zeitbereiche dar. Die einzelnen Zeitreihen werden bei der Speicherung sequenziell in sogenannte Shards unterteilt, wobei jeder Shard in einer einzelnen Datenbank gespeichert wird. Eine vorausschauenden Einrichtung verschiedener Shard-Spaces (4 Stunden, 1 Tag, 1 Woche etc.) ermöglicht es, das langsame Löschen von Zeitbereichen durch das einfache Löschen ganzer Shards also ganzer Datenbanken (drop database) zu kompensieren. Eine verteilte Speicherung der Shards auf verschiedenen Rechnerknoten die wiederum in verschiedenen Clustern organisiert sein können, ermöglicht eine Verteilung der Daten, die falls gewünscht auch redundant mittels Replikation auf verschiedene Knoten erfolgen kann. Die Verteilung der Daten auf verschiedene Rechnerknoten ermöglicht es auch die Berechnung von Aggregaten über Zeitfenster die unterhalb der Shardgröße liegen, zu verteilen und somit Lokalität der Daten und einen Performance-Vorteil zu erreichen. Auch hier ist es sinnvoll Shardgrößen vorausschauend zu planen. Die Anfragen an InfluxDB können mittels einer SQL-ähnlichen Anfragesprache über eine http-Schnittstelle formuliert werden. Es werden verschiedene Aggregatfunktionen bereitgestellt, die eine Ausgabe bspw. gruppiert nach Zeitintervallen für einen gesamten Zeitbereich erzeugen, wobei die Verwendung Regulärer Ausdrücke unterstützt wird:

```
select median(used) from /cpu\.*/
where time > now() - 4h group by time(5m)
```

Hier wird der Median des „used“-Wertes für alle 5-Minuten-Fenster der letzten 4 Stunden für alle CPUs berechnet und ausgegeben. Neben normalen Anfragen können auch sogenannte Continuous Queries eingerichtet werden, die etwa das einfache Downsampling von Messdaten ermöglichen:

[6]node.js - a cross-platform runtime environment for server-side and networking applications. http://nodejs.org/.

[7]RabbitMQ - Messaging that just works. http://www.rabbitmq.com.

[8]MongoDB - An open-source document database. http://www.mongodb.org/.

[9]RStudio - open source and enterprise-ready professional software for the R statistical computing environment. http://www.rstudio.com.

[10]Shiny - A web application framework for R. http://shiny.rstudio.com.

[11]Oracle. http://www.oracle.com.

[12]IBM Informix. http://www-01.ibm.com/software/data/informix/.

[13]PostgreSQL. http://www.postgresql.org/.

[14]OpenTSDB - Scalable Time Series Database. http://opentsdb.net/.

[15]KairosDB - Fast Scalable Time Series Database. https://code.google.com/p/kairosdb/.

[16]RRDB - Round Robin Database. http://oss.oetiker.ch/rrdtool/.

[17]InfluxDB - An open-source distributed time series database with no external dependencies. http://influxdb.com/.

[18]LevelDB - A fast and lightweight key-value database library by Google. http://code.google.com/p/leveldb/.

```
select count(name) from clicks
group by time(1h) into clicks.count.1h
```

InfluxDB befindet sich noch in einem frühen Stadium der Entwicklung und wird ständig weiterentwickelt. So ist etwa angekündigt, dass zukünftig bspw. das Speichern von Metadaten zu Zeitreihen (Einheiten, Abtastrate, etc.) oder auch die Implementierung nutzerdefinierter Aggregatfunktionen ermöglicht wird. InfluxDB ist ein für unsere Anwendung vielversprechendes Werkzeug, wobei jedoch abzuwarten bleibt, inwiefern es sich für den produktiven Einsatz eignet. Aus diesem Grund wird derzeit zusätzlich zu InfluxDB, MongoDB parallel als im Unternehmen bewährter Datenspeicher verwendet.

4. LÖSUNG IN PAGEBEAT

Im Projekt Pagebeat wurden verschiedene Lösungsansätze getestet, wobei die Praktikabilität beim Einsatz im Unternehmen, die schnelle Umsetzbarkeit sowie die freie Verfügbarkeit der eingesetzten Werkzeuge die entscheidende Rolle spielten.

4.1 Datenfluss

Der Datenfluss innerhalb der Gesamtarchitektur ist in Abbildung 1 dargestellt. Die Messdaten werden von einer Drohne[19] sowie Clientsimulatoren und Lasttestservern in äquidistanten Zeitabschnitten (meist 10 s) ermittelt. Die erhobenen Daten werden einem Loggingdienst per REST-Schnittstelle zur Verfügung gestellt und reihen sich in die Warteschlange eines Nachrichtenservers ein. Von dort aus werden sie ihrer Signatur entsprechend durch registrierte Analyse- bzw. Interpretationsprozesse verarbeitet, wobei die Validierung der eintreffenden Daten sowie die Zuordnung zu registrierten Analysefunktionen mittels einer Wissensbasis erfolgt. Ergebnisse werden wiederum als Nachricht zur Verfügung gestellt und falls vorgesehen persistent gespeichert. So in die Nachrichtenschlange gekommene Ergebnisse können nun weitere Analysen bzw. Interpretationen oder die Auslösung einer Benachrichtigung zur Folge haben. Der Daten Explorer ermöglicht eine Sichtung von Rohdaten und bereits in PageBeat integrierten Analyseergebnissen sowie Tests für zukünftige Analysefunktionen.

4.2 Wissensbasis

Die Wissenbasis bildet die Grundlage für die modular aufgebauten Analyse- und Interpretationsprozesse. Die Abbildung 2 dargestellten „ParameterValues“ repräsentieren die Messdaten und deren Eigenschaften wie Name, Beschreibung oder Einheit. ParameterValues können zu logischen Gruppen (Parameters) zusammengefasst werden (wie z.B. die ParameterValues: „system“, „load“, „iowait“ und „max“ zum Parameter „cpu“). Parameter sind mit Visualisierungskomponenten und Kundendaten sowie mit Analysen und Interpretationen verknüpft. Analysen und Interpretationen sind modular aufgebaut und bestehen jeweils aus Eingangs- und Ausgangsdaten (ParameterValues) sowie aus Verweisen auf den Programmcode. Weiterhin sind ihnen spezielle Methodenparameter zugeordnet. Hierbei handelt es sich etwa um Start und Ende eines Zeitfensters, Schwellenwerte oder andere Modellparameter. Die Wissensbasis ist mittels eines relationalem Schemas in MySQL abgebildet.

[19]Auf dem zu beobachtenden System installierter Agent zur Datenerhebung.

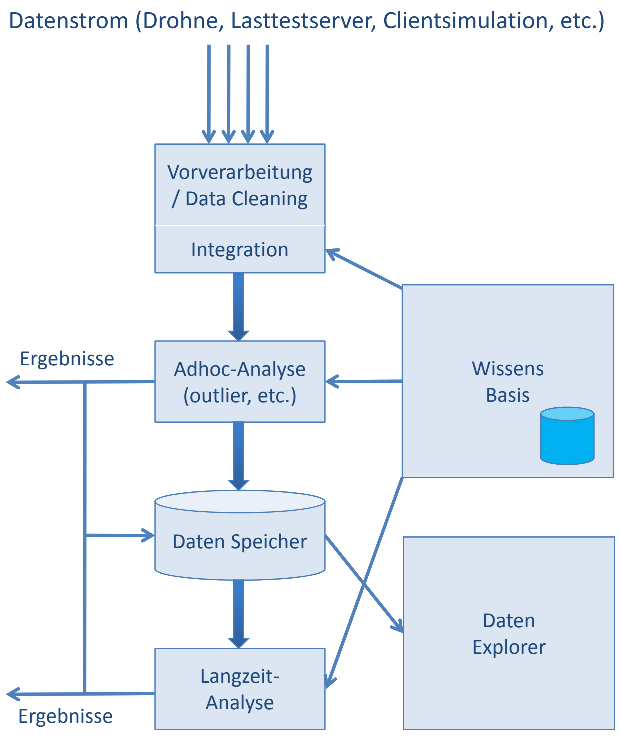

Datenstrom (Drohne, Lasttestserver, Clientsimulation, etc.)
Vorverarbeitung / Data Cleaning
Integration
Ergebnisse
Adhoc-Analyse (outlier, etc.)
Wissens Basis
Daten Speicher
Daten Explorer
Langzeit-Analyse
Ergebnisse

Abbildung 1: Datenfluss

4.3 Speicherung der Zeitreihen

Die Speicherung der Messdaten sowie Analyse- und Interpretationsergebnisse erfolgt zum Einen in der im Unternehmen bewährten, auf hochfrequente Schreibvorgänge optimierten schemafreien Datenbank MongoDB. Zum Anderen setzen wir mittlerweile parallel zu MongoDB auf InfluxDB. So kann z.B. über die in InluxDB zur Verfügung stehenden Continious Queries ein automatisches Downsampling und somit eine Datenreduktion der im 10 Sekunden Takt erhobenen Daten erfolgen. Das Downsampling erfolgt derzeit durch die Berechnung der Mittelwerte von Zeitfenstern einer Länge von 1 Minute bis hin zu einem Tag und generiert somit automatisch unterschiedliche zeitliche Auflösungen für alle Messwerte. Außerdem stellt die SQL ähnliche Anfragesprache von InfluxDB eine Vielzahl von für die statistische Auswertung hilfreichen Aggregatfunktionen (min, max, mean, median, stddev, percentile, histogramm, etc.) zur Verfügung. Weiterhin soll es zukünftig möglich sein benutzerdefinierte Funktionen mit eigener Analysefunktionalität (etwa Autokorrelation, Kreuzkorrelation, Vorhersage, etc.) auf Datenbankebene umzusetzen oder auch das automatische Zusammenführen verschiedener Zeitserien anhand eines Timestamp-Attributs durchzuführen. Dies würde schon auf Datenbankebene eine zeitreihenübergreifende Analyse (bspw. Korrelation) unterstützen und senkt den Reimplentierungsaufwand von R Funktionalität aus der Datenexplorationsphase. Da herkömmliche Datenbanken nicht die hohe Performance bzgl. Schreibzugriffen erreichen und kaum auf Zeitreihen spezialisierte Anfragen unterstützen, scheint InfluxDB ein geeigneter Kandidat für den Einsatz innerhalb PageBeats zu sein.

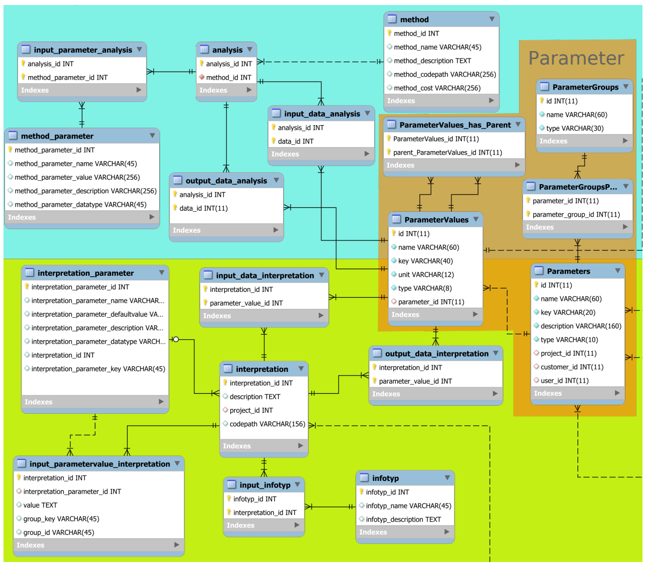


Abbildung 2: Ausschnitt Schema Wissensbasis

4.4 Datenexploration

Die Datenexploration soll dazu dienen, Administratoren und auch Endnutzern die Möglichkeit zu geben, die für sie relevanten Daten mit den richtigen Werkzeugen zu analysieren. Während der Entwicklung nutzen wir die Datenexploration als Werkzeug zur Ermittlung relevanter Analysemethoden und zur Evaluierung sowie Visualisierung der Datenströme. Abbildung 3 zeigt eine einfache Nutzerschnittstelle umgesetzt mit Shiny zur Datenauswertung mittels R mit Zugriff auf unterschiedliche Datenbanken, InfluxDB und MongoDB. Verschiedene Parameter zur Auswahl des Zeitraumes, der Analysefunktion und deren Parameter sowie Visualisierungsparameter.
Hier sind durchschnittliche CPU-Nutzung und durchschnittliche Plattenzugriffszeiten aus einer Auswahl aus 10 Zeitserien dargestellt. Mittels unterem Interaktionselement lassen sich Intervalle selektieren und die Granularität anpassen. Mit ähnlichen Visualisierungsmethoden lassen sich auch Autokorrelationsanalysen visualisieren, siehe Abbildung 4.

4.5 Analyse und Interpretation

Analysen sind Basisoperationen wie die Berechnung von Mittelwert, Median, Standardabweichung, Autokorrelation u.a. deren Ergebnisse falls nötig persistent gespeichert werden oder direkt anderen Verarbeitungsschritten als Eingabe übergeben werden können. Die Spezifizierung der Analysefunktionen erfolgt in der Wissensbasis, die eigentliche Implementierung ist möglichst nahe an den zu analysierenden Daten, wenn möglich unter Verwendung von Aggregat- oder benutzerdefinierten Funktionen des Datenbanksystems, umzusetzen. Wissensbasis und Analyse sind hierzu mittels eines „method_codepath“ verknüpft.
Interpretationen funktionieren analog zur Analyse bilden jedoch Berechnungsvorschriften etwa für den Gesamtindex (Pagebeat-Faktor) des Systems bzw. einzelner Teilsysteme ab, in dem sie z.B. Analyseergebnisse einzelner Zeitreihen gewichtet zusammenführen. Weiterhin besitzen Interpretationen einen Infotyp, welcher der nutzerspezifischen Aufbereitung von Ergebnissen dient. Abbildung 5 zeigt etwa die Darstellung aggregierter Parameter in Ampelform (rot = kritisch, gelb = Warnung, grün = normal, blau = optimal) was schnell einen Eindruck über den Zustand verschiedener Systemparameter ermöglicht.

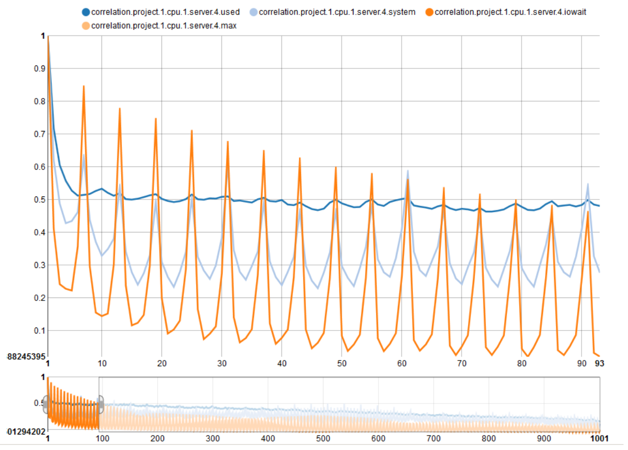


Abbildung 4: Autokorrelation

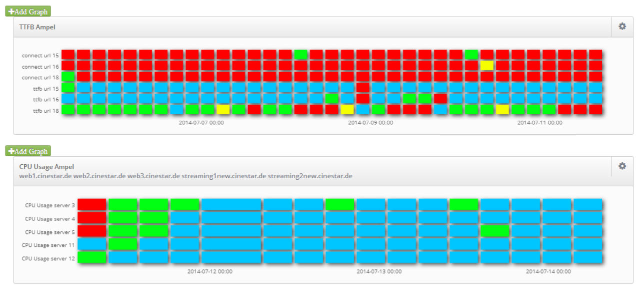
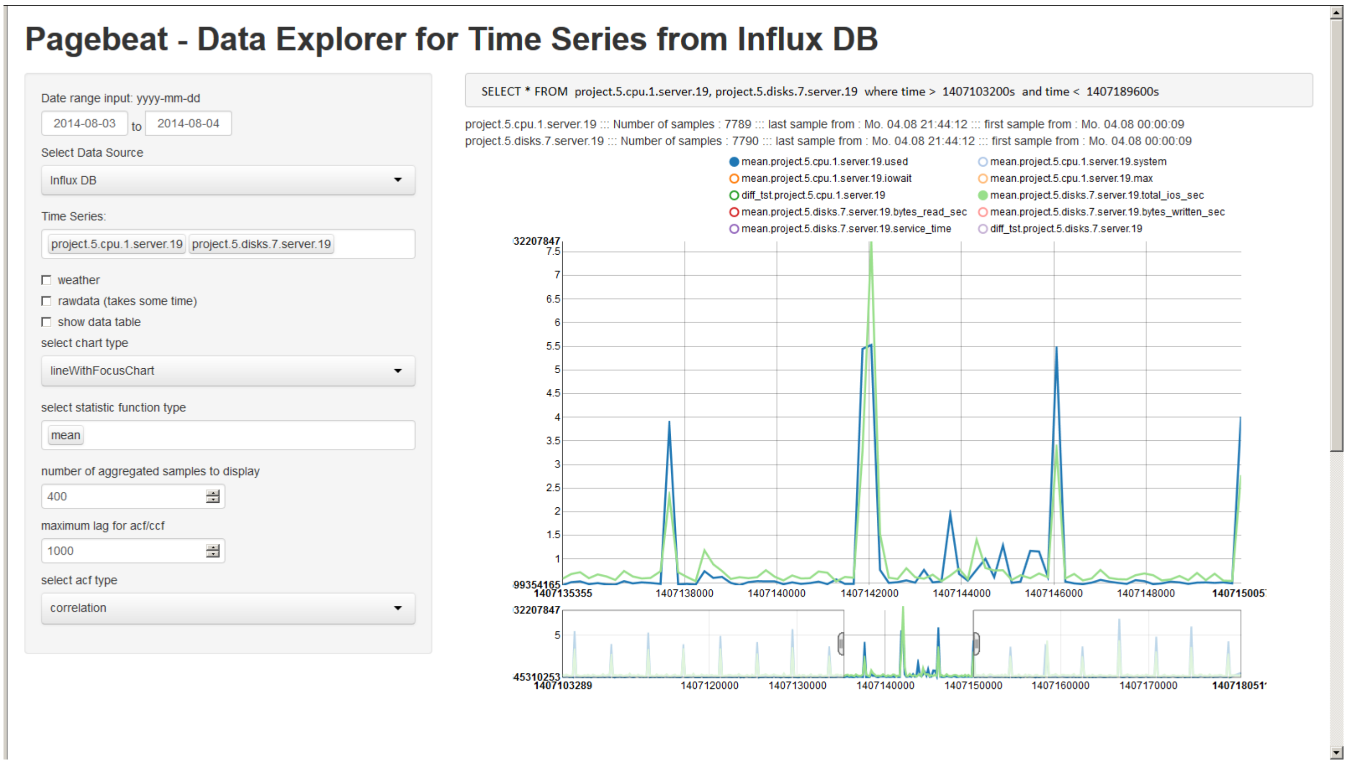


Abbildung 5: Ampel

Analysefunktionalität die über Aggregationen auf Datenbankebene hinausgehen wird von uns in einer Experimentalumgebung umgesetzt und evaluiert. Diese basiert auf R. So stehen eine Vielzahl statistischer Analysemethoden und Methoden zur Aufbereitung komplexer Datenstrukturen in Form von R Paketen zur Verfügung. Darüber hinaus ermöglicht das R-Paket „Shiny Server“ die komfortable Bereitstellung von R Funktionalität für das Web. Ein wesentlicher Teil unser Experimentalumgebung ist der Pagebeat Data Explorer (siehe Abbildung 3). Dieser basiert auf den genannten Techniken und ermöglicht die Sichtung der erfassten Rohdaten oder das „Spielen“ mit Analysemethoden und Vorhersagemodellen.

5. ZUSAMMENFASSUNG UND AUSBLICK

Pagebeat ist ein Projekt, bei dem es insbesondere auf eine performante Speicherung und schnelle Adhoc-Auswertung der Daten ankommt. Dazu wurden verschiedene Lösungsansätze betrachtet und die favorisierte Lösung auf Basis von InfluxDB und R beschrieben.
Die konzeptionelle Phase ist abgeschlossen, die Projektinfrastruktur umgesetzt und erste Analysemethoden wie Aus-



Pagebeat - Data Explorer for Time Series from Influx DB

Date range input: yyyy-mm-dd
2014-08-03 to 2014-08-04
Select Data Source
Influx DB
Time Series:
project.5.cpu.1.server.19 project.5.disks.7.server.19
☐ weather
☐ rawdata (takes some time)
☐ show data table
select chart type
lineWithFocusChart
select statistic function type
mean
number of aggregated samples to display
400
maximum lag for acf/ccf
1000
select acf type
correlation

```
SELECT * FROM project.5.cpu.1.server.19, project.5.disks.7.server.19 where time > 1407103200s and time < 1407189600s
```

project.5.cpu.1.server.19 ::: Number of samples : 7789 ::: last sample from : Mo. 04.08 21:44:12 ::: first sample from : Mo. 04.08 00:00:09
project.5.disks.7.server.19 ::: Number of samples : 7790 ::: last sample from : Mo. 04.08 21:44:12 ::: first sample from : Mo. 04.08 00:00:09

● mean.project.5.cpu.1.server.19.used
○ mean.project.5.cpu.1.server.19.system
○ mean.project.5.cpu.1.server.19.iowait
○ mean.project.5.cpu.1.server.19.max
○ diff_tst.project.5.cpu.1.server.19
● mean.project.5.disks.7.server.19.total_ios_sec
○ mean.project.5.disks.7.server.19.bytes_read_sec
○ mean.project.5.disks.7.server.19.bytes_written_sec
○ mean.project.5.disks.7.server.19.service_time
○ diff_tst.project.5.disks.7.server.19
32207847
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
99354165
1407135355
1407138000
1407140000
1407142000
1407144000
1407146000
1407148000
140715005
32207847
5
45310253
1407103289
1407120000
1407130000
1407140000
1407150000
1407160000
1407170000
140718051

Abbildung 3: Daten

reißererkennung oder Autokorrelation wurden ausprobiert. Derzeit beschäftigen wir uns mit den Möglichkeiten einer Vorhersage von Zeitreihenwerten. Dazu werden Ergebnisse der Autokorrelationsanalyse zur Identifikation von Abhängigkeiten innerhalb von Zeitreihen verwendet um die Qualität von Vorhersagen abschätzen zu können. Weiterhin ist geplant Analysen näher an der Datenbank auszuführen um Datenlokalität zu unterstützen.

6. REFERENCES

[1] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom. Stream: The stanford data stream management system. Technical Report 2004-20, Stanford InfoLab, 2004.

[2] A. Arasu, S. Babu, and J. Widom. The cql continuous query language: Semantic foundations and query execution. Technical Report 2003-67, Stanford InfoLab, 2003.

[3] K. Chinda and R. Vijay. Informix timeseries solution. `http://www.ibm.com/developerworks/data/library/techarticle/dm-1203timeseries`, 2012.

[4] O. Corporation. R technologies from oracle. `http://www.oracle.com/technetwork/topics/bigdata/r-offerings-1566363.html`, 2014.

[5] EsperTech. Esper. `http://esper.codehaus.org`, 2014.

[6] U. Fischer, L. Dannecker, L. Siksnys, F. Rosenthal, M. Boehm, and W. Lehner. Towards integrated data analytics: Time series forecasting in dbms. *Datenbank-Spektrum*, 13(1):45–53, 2013.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

Databases under the Partial Closed-world Assumption: A Survey

Simon Razniewski
Free University of Bozen-Bolzano
Dominikanerplatz 3
39100 Bozen, Italy
razniewski@inf.unibz.it

Werner Nutt
Free University of Bozen-Bolzano
Dominikanerplatz 3
39100 Bozen, Italy
nutt@inf.unibz.it

ABSTRACT

Databases are traditionally considered either under the closed-world or the open-world assumption. In some scenarios however a middle ground, the partial closed-world assumption, is needed, which has received less attention so far.

In this survey we review foundational and work on the partial closed-world assumption and then discuss work done in our group in recent years on various aspects of reasoning over databases under this assumption.

We first discuss the conceptual foundations of this assumption. We then list the main decision problems and the known results. Finally, we discuss implementational approaches and extensions.

1. INTRODUCTION

Data completeness is an important aspect of data quality. Traditionally, it is assumed that a database reflects exactly the state of affairs in an application domain, that is, a fact that is true in the real world is stored in the database, and a fact that is missing in the database does not hold in the real world. This is known as the closed-world assumption (CWA). Later approaches have discussed the meaning of databases that are missing facts that hold in the real world and thus are incomplete. This is called the open-world assumption (OWA) [16, 7].

A middle view, which we call the partial closed-world assumption (PCWA), has received less attention until recently. Under the PCWA, some parts of the database are assumed to be closed (complete), while others are assumed to be open (possibly incomplete). So far, the former parts were specified using completeness statements, while the latter parts are the complement of the complete parts.

Example. As an example, consider a problem arising in the management of school data in the province of Bolzano, Italy, which motivated the technical work reported here. The IT department of the provincial school administration runs a database for storing school data, which is maintained in a decentralized manner, as each school is responsible for its own data. Since there are numerous schools in this province, the overall database is notoriously incomplete. However, periodically the statistics department of the province queries the school database to generate statistical reports. These statistics are the basis for administrative decisions such as the opening and closing of classes, the assignment of teachers to schools and others. It is therefore important that these statistics are correct. Therefore, the IT department is interested in finding out which data has to be complete in order to guarantee correctness of the statistics, and on which basis the guarantees can be given.

Broadly, we investigated the following research questions:

1. How to describe complete parts of a database?

2. How to find out, whether a query answer over a partially closed database is complete?

3. If a query answer is not complete, how to find out which kind of data can be missing, and which similar queries are complete?

Work Overview. The first work on the PCWA is from Motro [10]. He used queries to describe complete parts and introduced the problem of inferring the completeness of other queries (QC) from such completeness statements. Later work by Halevy [8] introduced tuple-generating dependencies or table completeness (TC) statements for specification of complete parts. A detailed complexity study of TC-QC entailment was done by Razniewski and Nutt [13].

Later work by Razniewski and Nutt has focussed on databases with null values [12] and geographic databases [14].

There has also been work on RDF data [3]. Savkovic et al. [18, 17] have focussed on implementation techniques, leveraging especially on logic programming.

Also the derivation of completeness from data-aware business process descriptions has been discussed [15].

Current work is focussing on reasoning wrt. database instances and on queries with negation [4].

Outline. This paper is structured as follows. In Section 2, we discuss conceptual foundations, in particular the partial closed-world assumption. In Section 3 we present main reasoning problems in this framework and known results. Section 4 discusses implementation techniques. Section 5 presents extension and Section 6 discusses current work and open problems.

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

2. CONCEPTUAL FOUNDATIONS

2.1 Standard Definitions

In the following, we fix our notation for standard concepts from database theory. We assume a set of relation symbols Σ, the *signature*. A *database instance* D is a finite set of ground atoms with relation symbols from Σ. For a relation symbol $R \in \Sigma$ we write $R(D)$ to denote the interpretation of R in D, that is, the set of atoms in D with relation symbol R. A *condition* G is a set of atoms using relations from Σ and possibly the comparison predicates $<$ and $\leq$. As common, we write a condition as a sequence of atoms, separated by commas. A condition is *safe* if each of its variables occurs in a relational atom. A *conjunctive query* is written in the form $Q(\bar{s}) :- B$, where B is a safe condition, $\bar{s}$ is a vector of terms, and every variable in $\bar{s}$ occurs in B. We often refer to the entire query by the symbol Q. As usual, we call $Q(\bar{s})$ the *head*, B the *body*, the variables in $\bar{s}$ the *distinguished variables*, and the remaining variables in B the *nondistinguished variables* of Q. We generically use the symbol L for the subcondition of B containing the relational atoms and M for the subcondition containing the comparisons. If B contains no comparisons, then Q is a *relational conjunctive query*.

The result of evaluating Q over a database instance D is denoted as $Q(D)$. Containment and equivalence of queries are defined as usual. A conjunctive query is *minimal* if no relational atom can be removed from its body without leading to a non-equivalent query.

2.2 Running Example

For our examples throughout the paper, we will use a drastically simplified extract taken from the schema of the Bolzano school database, containing the following two tables:

- *student(name, level, code),*
- *person(name, gender).*

The table *student* contains records about students, that is, their names and the level and code of the class we are in. The table *person* contains records about persons (students, teachers, etc.), that is, their names and genders.

2.3 Completeness

Open and closed world semantics were first discussed by Reiter in [16], where he formalized earlier work on negation as failure [2] from a database point of view. The closed-world assumption corresponds to the assumption that the whole database is complete, while the open-world assumption corresponds to the assumption that nothing is known about the completeness of the database.

Partial Database. The first and very basic concept is that of a partially complete database or partial database [10]. A database can only be incomplete with respect to another database that is considered to be complete. So we model a partial database as a pair of database instances: one instance that describes the complete state, and another instance that describes the actual, possibly incomplete state. Formally, a *partial database* is a pair $\mathcal{D} = (D^i, D^a)$ of two database instances D^i and D^a such that $D^a \subseteq D^i$. In the style of [8], we call D^i the *ideal* database, and D^a the *available* database. The requirement that D^a is included in D^i formalizes the intuition that the available database contains no more information than the ideal one.

Example 1. *Consider a partial database $\mathcal{D}_S$ for a school with two students, Hans and Maria, and one teacher, Carlo, as follows:*

$$\begin{aligned} D^i_S &= \{\textit{student(Hans, 3, A)}, \textit{student(Maria, 5, C)}, \\ &\qquad \textit{person(Hans, male)}, \textit{person(Maria, female)}, \\ &\qquad \textit{person(Carlo, male)}\}, \\ D^a_S &= D^i_S \setminus \{\textit{person(Carlo, male)}, \textit{student(Maria, 5, C)}\}, \end{aligned}$$

that is, the available database misses the facts that Maria is a student and that Carlo is a person. □

Next, we define statements to express that parts of the information in D^a are complete with regard to the ideal database D^i. We distinguish query completeness and table completeness statements.

Query Completeness. For a query Q, the *query completeness* statement *Compl*(Q) says that Q can be answered completely over the available database. Formally, *Compl*(Q) is *satisfied* by a partial database $\mathcal{D}$, denoted as $\mathcal{D} \models \mathit{Compl}(Q)$, if $Q(D^a) = Q(D^i)$.

Example 2. *Consider the above defined partial database $\mathcal{D}_S$ and the query*

$$Q_1(n) :- \mathit{student}(n, l, c), \mathit{person}(n, \text{'male'}),$$

asking for all male students. Over both, the available database D^a_S and the ideal database D^i_S, this query returns exactly Hans. Thus, $\mathcal{D}_S$ satisfies the query completeness statement for Q_1, that is,

$$\mathcal{D}_S \models \mathit{Compl}(Q_1). \quad \square$$

Abiteboul et al. [1] introduced the notion of certain and possible answers over databases under the open-world assumption. Query completeness can also be seen as a relation between certain and possible answers: A query over a partially complete database is complete, if the certain and the possible answers coincide.

Table completeness. A table completeness (TC) statement allows one to say that a certain part of a relation is complete, without requiring the completeness of other parts of the database [8]. It has two components, a relation R and a condition G. Intuitively, it says that all tuples of the ideal relation R that satisfy condition G in the ideal database are also present in the available relation R.

Formally, let $R(\bar{s})$ be an R-atom and let G be a condition such that $R(\bar{s}), G$ is safe. We remark that G can contain relational and built-in atoms and that we do not make any safety assumptions about G alone. Then $\mathit{Compl}(R(\bar{s}); G)$ is a *table completeness statement*. It has an *associated query*, which is defined as $Q_{R(\bar{s});G}(\bar{s}) :- R(\bar{s}), G$. The statement is satisfied by $\mathcal{D} = (D^i, D^a)$, written $\mathcal{D} \models \mathit{Compl}(R(\bar{s}); G)$, if $Q_{R(\bar{s});G}(D^i) \subseteq R(D^a)$. Note that the ideal instance $\hat{D}$ is used to determine those tuples in the ideal version $R(D^i)$ that satisfy G and that the statement is satisfied if these tuples are present in the available version $R(D^a)$. In the sequel, we will denote a TC statement generically as C and refer to the associated query simply as Q_C.

If we introduce different schemas Σ^i and Σ^a for the ideal and the available database, respectively, we can view the TC statement $C = \mathit{Compl}(R(\bar{s}); G)$ equivalently as the TGD (= tuple-generating dependency) $\delta_C \colon R^i(\bar{s}), G^i \rightarrow R^a(\bar{s})$ from Σ^i to

Σ^a. It is straightforward to see that a partial database satisfies the TC statement C if and only if it satisfies the TGD δ_C.

The view of TC statements is especially useful for implementations.

Example 3. *In the partial database $\mathcal{D}_S$ defined above, we can observe that in the available relation person, the teacher Carlo is missing, while all students are present. Thus, person is complete for all students. The available relation student contains Hans, who is the only male student. Thus, student is complete for all male persons. Formally, these two observations can be written as table completeness statements:*

$$C_1 = Compl(person(n,g); student(n,l,c)),$$
$$C_2 = Compl(student(n,l,c); person(n, 'male')),$$

which, as seen, are satisfied by the partial database $\mathcal{D}_S$. □

One can prove that table completeness cannot be expressed by query completeness statements, because the latter require completeness of the relevant parts of all the tables that appear in the statement, while the former only talks about the completeness of a single table.

Example 4. *As an illustration, consider the table completeness statement C_1 that states that person is complete for all students. The corresponding query Q_{C_1} that asks for all persons that are students is*

$$Q_{C_1}(n,g) :- person(n,g), student(n,l,c).$$

Evaluating Q_{C_1} over D^i_S gives the result {Hans, Maria}. However, evaluating it over D^a_S returns only {Hans}. Thus, $\mathcal{D}_S$ does not satisfy the completeness of the query Q_{C_1} although it satisfies the table completeness statement C_1. □

Reasoning. As usual, a set $\mathcal{S}_1$ of TC- or QC-statements *entails* another set $\mathcal{S}_2$ (we write $\mathcal{S}_1 \models \mathcal{S}_2$) if every partial database that satisfies all elements of $\mathcal{S}_1$ also satisfies all elements of $\mathcal{S}_2$.

Example 5. *Consider the query $Q(n) :- student(n,7,c)$, $person(n,'male')$ that asks for all male students in level 7. The TC statements C_1 and C_2 entail completeness of this query, because we ensure that all persons that are students and all male students are in the database. Note that these are not the minimal preconditions, as it would be enough to only have male persons in the database who are student in level 7, and students in level 7, who are male persons.*

While TC statements are a natural way to describe completeness of available data ("These parts of the data are complete"), QC statements capture requirements for data quality ("For these queries we need complete answers"). Thus, checking whether a set of TC statements entails a set of QC statements (TC-QC entailment) is the practically most relevant inference. Checking TC-TC entailment is useful when managing sets of TC statements. Moreover, as we will show later on, TC-QC entailment for aggregate queries with count and sum can be reduced to TC-TC entailment for non-aggregate queries. If completeness guarantees are given in terms of query completeness, also QC-QC entailment is of interest.

3. CHARACTERIZATIONS AND DECISION PROCEDURES

Motro [10] introduced the notion of partially incomplete and incorrect databases as databases that can both miss facts that hold in the real world or contain facts that do not hold there. He described partial completeness in terms of *query completeness* (QC) statements, which express that the answer of a query is complete. The query completeness statements express that to some parts of the database the closed-world assumption applies, while for the rest of the database, the open-world assumption applies. He studied how the completeness of a given query can be deduced from the completeness of other queries, which is QC-QC entailment. His solution was based on rewriting queries using views: to infer that a given query is complete whenever a set of other queries are complete, he would search for a conjunctive rewriting in terms of the complete queries. This solution is correct, but not complete, as later results on query determinacy show: the given query may be complete although no conjunctive rewriting exists.

While Levy et al. could show that rewritability of conjunctive queries as conjunctive queries is decidable [9], general rewritability of conjunctive queries by conjunctive queries is still open: An extensive discussion on that issue was published in 2005 by Segoufin and Vianu where it is shown that it is possible that conjunctive queries can be rewritten using other conjunctive queries, but the rewriting is not a conjunctive query [19]. They also introduced the notion of query determinacy, which for conjunctive queries implies second order rewritability. The decidability of query determinacy for conjunctive queries is an open problem to date.

Halevy [8] suggested local completeness statements, which we, for a better distinction from the QC statements, call table completeness (TC) statements, as an alternate formalism for expressing partial completeness of an incomplete database. These statements allow one to express completeness of parts of relations independent from the completeness of other parts of the database. The main problem he addressed was how to derive query completeness from table completeness (TC-QC). He reduced TC-QC to the problem of queries independent of updates (QIU) [5]. However, this reduction introduces negation, and thus, except for trivial cases, generates QIU instances for which no decision procedures are known. As a consequence, the decidability of TC-QC remained largely open. Moreover, he demonstrated that by taking into account the concrete database instance and exploiting the key constraints over it, additional queries can be shown to be complete.

Razniewski and Nutt provided decision procedures for TC-QC in [13]. They showed that for queries under bag semantics and for minimal queries under set semantics, weakest preconditions for query completeness can be expressed in terms of table completeness statements, which allow to reduce TC-QC entailment to TC-TC entailment.

For the problem of TC-TC entailment, they showed that it is equivalent to query containment.

For QC-QC entailment, they showed that the problem is decidable for queries under bag semantics.

For aggregate queries, they showed that for the aggregate functions SUM and COUNT, TC-QC has the same complexity as TC-QC for nonaggregate queries under bag semantics. For the aggregate functions MIN and MAX, they showed that

Problem	Work by	Results
QC-QC	Motro 1989	Query rewritability is a sufficient condition for QC-QC^s
	Razniewski/Nutt 2011	QC-QC^b is equivalent to query containment
TC-TC	Razniewski/Nutt 2011	TC-TC is equivalent to query containment
TC-QC	Levy 1996	Decision procedure for trivial cases
	Razniewski/Nutt 2011	TC-QC^b is equivalent to TC-TC, TC-QC^s is equivalent to TC-TC up to asymmetric cases
	Razniewski/Nutt 2012	Decision procedures for TC-QC^s over databases with nulls

Table 1: Main results

TC-QC has the same complexity as TC-QC for nonaggregate queries under set semantics.

For reasoning wrt. a database instance, they showed that TC-QC becomes computationally harder than without an instance, while QC-QC surprisingly becomes solvable, whereas without an instance, decidability is open.

In [12], Nutt and Razniewski discussed TC-QC entailment reasoning over databases that contain null values. Null values as used in SQL are ambiguous, as they can indicate either that no attribute value exists or that a value exists, but is unknown. Nutt and Razniewski studied completeness reasoning for both interpretations, and showed that when allowing both interpretations at the same time, it becomes necessary to syntactically distinguish between different kinds of null values. They presented an encoding for doing that in standard SQL databases. With this technique, any SQL DBMS evaluates complete queries correctly with respect to the different meanings that null values can carry.

The main results are summarized in Table 1.

4. IMPLEMENTATION TECHNIQUES

Systems for reasoning can be developed from scratch, however it may be useful to implement them using existing technology as far as possible. So far, it was investigated how completeness reasoning can be reduced to answer set programming, in particular using the DLV system.

The MAGIK system developed by Savkovic et al. [18] demonstrates how to use meta-information about the completeness of a database to assess the quality of the answers returned by a query. The system holds table-completeness (TC) statements, by which one can express that a table is partially complete, that is, it contains all facts about some aspect of the domain.

Given a query, MAGIK determines from such meta-information whether the database contains sufficient data for the query answer to be complete (TC-QC entailment). If, according to the TC statements, the database content is not sufficient for a complete answer, MAGIK explains which further TC statements are needed to guarantee completeness.

MAGIK extends and complements theoretical work on modeling and reasoning about data completeness by providing the first implementation of a reasoner. The reasoner operates by translating completeness reasoning tasks into logic programs, which are executed by an answer set engine.

In [17], Savkovic et al. present an extension to MAGIK that computes for a query that may be incomplete, complete approximations from above and from below. With this extension, they show how to reformulate the original query in such a way that answers are guaranteed to be complete. If there exists a more general complete query, there is a unique most specific one, which is found. If there exists a more specific complete query, there may even be infinitely many. In this case, the least specific specializations whose size is bounded by a threshold provided by the user is found. Generalizations are computed by a fixpoint iteration, employing an answer set programming engine. Specializations are found leveraging unification from logic programming.

5. EXTENSIONS AND APPLICATIONS SCENARIOS

Complete generalizations and specializations. When a query is not guaranteed to be complete, it may be interesting to know which similar queries are complete. For instance, when a query for all students in level 5 is not complete, it may still be the case that the query for students in classes 5b and 5c is complete. Such information is especially interesting for interaction with a completeness reasoning system. In [11], Savkovic et al. defined the notion of most general complete specialization and the most specific comple generalization, and discussed techniques to find those.

Completeness over Business Processes. In many applications, data is managed via well documented processes. If information about such processes exists, one can draw conclusions about completeness as well. In [15], Razniewski et al. presented a formalization of so-called *quality-aware processes* that create data in the real world and store it in the company's information system possibly at a later point. They then showed how one can check the completeness of database queries in a certain state of the process or after the execution of a sequence of actions, by leveraging on query containment, a well-studied problem in database theory. Finally, they showed how the results can be extended to the more expressive formalism of colored Petri nets.

Spatial Data. Volunteered geographical information systems are gaining popularity. The most established one is OpenStreetMap (OSM), but also classical commercial map services such as Google Maps now allow users to take part in the content creation.

Assessing the quality of spatial information is essential for making informed decisions based on the data, and particularly challenging when the data is provided in a decentralized, crowd-based manner. In [14], Razniewski and Nutt showed how information about the completeness of features in certain regions can be used to annotate query answers with completeness information. They provided a characterization of the necessary reasoning and show that when taking into account the available database, more completeness can be derived. OSM already contains some completeness statements, which are originally intended for coordination among the editors of the map. A contribution was also to show that these statements are not only useful for the producers of the data but also for the consumers.

RDF Data. With thousands of RDF data sources today available on the Web, covering disparate and possibly overlapping knowledge domains, the problem of providing high-level descriptions (in the form of metadata) of their content becomes crucial. In [3], Darari et al. discussed reasoning about the completeness of semantic web data sources. They showed how the previous theory can be adapted for RDF data sources, what peculiarities the SPARQL query language offers and how completeness statements themselves can be expressed in RDF.

They also discussed the foundation for the expression of completeness statements about RDF data sources. This allows to complement with *qualitative* descriptions about completeness the existing proposals like VOID that mainly deal with *quantitative* descriptions. The second aspect of their work is to show that completeness statements can be useful for the semantic web in practice. On the theoretical side, they provide a formalization of completeness for RDF data sources and techniques to reason about the completeness of query answers. From the practical side, completeness statements can be easily embedded in current descriptions of data sources and thus readily used. The results on RDF data have been implemented by Darari et al. in a demo system called CORNER [6].

6. CURRENT WORK

In this section we list problems that our group is currently working on.

6.1 SPARQL Queries with Negation

RDF data is often treated as incomplete, following the Open-World Assumption. On the other hand, SPARQL, the standard query language over RDF, usually follows the Closed-World Assumption, assuming RDF data to be complete. What then happens is the semantic gap between RDF and SPARQL. In current work, Darari et al. [4] address how to close the semantic gap between RDF and SPARQL, in terms of certain answers and possible answers using completeness statements. Table 2 shows current results for the relations between query answers, certain answers and possible answers for queries with negation. The queries are assumed to be of the form $Q(\bar{s}) :- P^+, \neg P^-$, where P^+ is the positive part and P^- is the negative part. Then we use letters C and N to indicate which parts are complete. E.g. $Q(\bar{s}) :- N, \neg C$ indicates that the positive part is not complete and the negative part is complete. As the table shows, depending on the complete parts, the query answer may either be equal to the possible answers, to the certain answers, both, or none.

Completeness P Pattern	Relationship between Certain Answers, Query Answers, and Possible Answers
$Q :- C$	$CA = QA = PA$
$Q :- N$	$CA = QA \subseteq PA = inf$
$Q :- N, \neg N$	$\emptyset = CA \subseteq QA \subseteq PA = inf$
$Q :- C, \neg C$	$CA = QA = PA$
$Q :- N, \neg C$	$CA = QA \subseteq PA = inf$
$Q :- C, \neg N$	$\emptyset = CA \subseteq QA = PA$

Table 2: Relation between query result, certain answers and possible answers for queries with negation. The arguments of Q are irrelevant and therefore omitted.

Note that the above results hold for conjunctive queries in general, and thus do not only apply to SPARQL but also to other query languages with negation, such as SQL.

6.2 Instance Reasoning

Another line of current work concerns completeness reasoning wrt. a database instance. We are currently looking into completeness statements which are simpler than TC statements in the sense that we do not contain any joins. For such statements, reasoning is still exponential in the size of the database schema, but experimental results suggest that in use cases, the reasoning is feasible. A challenge is however to develop a procedure which is algorithmically complete.

7. ACKNOWLEDGEMENT

We thank our collaborators Fariz Darari, Flip Korn, Paramita Mirza, Marco Montali, Sergey Paramonov, Giuseppe Pirró, Radityo Eko Prasojo, Ognjen Savkovic and Divesh Srivastava.

This work has been partially supported by the project "MAGIC: Managing Completeness of Data" funded by the province of Bozen-Bolzano.

8. REFERENCES

[1] S. Abiteboul, P.C. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. In *Proc. SIGMOD*, pages 34–48, 1987.

[2] Keith L Clark. Negation as failure. In *Logic and data bases*, pages 293–322. Springer, 1978.

[3] Fariz Darari, Werner Nutt, Giuseppe Pirrò, and Simon Razniewski. Completeness statements about RDF data sources and their use for query answering. In *International Semantic Web Conference (1)*, pages 66–83, 2013.

[4] Fariz Darari, Simon Razniewski, and Werner Nutt. Bridging the semantic gap between RDF and SPARQL using completeness statements. *ISWC*, 2013.

[5] Ch. Elkan. Independence of logic database queries and updates. In *Proc. PODS*, pages 154–160, 1990.

[6] Radityo Eko Prasojo Fariz Darari and Werner Nutt. CORNER: A completeness reasoner for the semantic web (poster). *ESWC*, 2013.

[7] T. Imieliński and W. Lipski, Jr. Incomplete information in relational databases. *J. ACM*, 31:761–791, 1984.

[8] Alon Y. Levy. Obtaining complete answers from incomplete databases. In *Proceedings of the International Conference on Very Large Data Bases*, pages 402–412, 1996.

[9] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *PODS*, pages 95–104, 1995.

[10] A. Motro. Integrity = Validity + Completeness. *ACM TODS*, 14(4):480–502, 1989.

[11] Werner Nutt, Sergey Paramonov, and Ognjen Savkovic. An ASP approach to query completeness reasoning. *TPLP*, 13(4-5-Online-Supplement), 2013.

[12] Werner Nutt and Simon Razniewski. Completeness of queries over SQL databases. In *CIKM*, pages 902–911, 2012.

[13] S. Razniewski and W. Nutt. Completeness of queries over incomplete databases. In *VLDB*, 2011.

[14] S. Razniewski and W. Nutt. Assessing the completeness of geographical data (short paper). In *BNCOD*, 2013.

[15] Simon Razniewski, Marco Montali, and Werner Nutt. Verification of query completeness over processes. In *BPM*, pages 155–170, 2013.

[16] Raymond Reiter. On closed world data bases. In *Logic and Data Bases*, pages 55–76, 1977.

[17] Ognjen Savkovic, Paramita Mirza, Sergey Paramonov, and Werner Nutt. Magik: managing completeness of data. In *CIKM*, pages 2725–2727, 2012.

[18] Ognjen Savkovic, Paramita Mirza, Alex Tomasi, and Werner Nutt. Complete approximations of incomplete queries. *PVLDB*, 6(12):1378–1381, 2013.

[19] L. Segoufin and V. Vianu. Views and queries: Determinacy and rewriting. In *Proc. PODS*, pages 49–60, 2005.

Towards Semantic Recommendation of Biodiversity Datasets based on Linked Open Data

Felicitas Löffler
Dept. of Mathematics and Computer Science
Friedrich Schiller University
Jena, Germany

Bahar Sateli
Semantic Software Lab
Dept. of Computer Science and Software Engineering
Concordia University
Montréal, Canada

René Witte
Semantic Software Lab
Dept. of Computer Science and Software Engineering
Concordia University
Montréal, Canada

Birgitta König-Ries
Friedrich Schiller University
Jena, Germany and
German Centre for Integrative Biodiversity Research (iDiv)
Halle-Jena-Leipzig, Germany

ABSTRACT

Conventional content-based filtering methods recommend documents based on extracted keywords. They calculate the similarity between keywords and user interests and return a list of matching documents. In the long run, this approach often leads to overspecialization and fewer new entries with respect to a user's preferences. Here, we propose a semantic recommender system using Linked Open Data for the user profile and adding semantic annotations to the index. Linked Open Data allows recommendations beyond the content domain and supports the detection of new information. One research area with a strong need for the discovery of new information is biodiversity. Due to their heterogeneity, the exploration of biodiversity data requires interdisciplinary collaboration. Personalization, in particular in recommender systems, can help to link the individual disciplines in biodiversity research and to discover relevant documents and datasets from various sources. We developed a first prototype for our semantic recommender system in this field, where a multitude of existing vocabularies facilitate our approach.

Categories and Subject Descriptors

H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval; H.3.5 [**Information Storage And Retrieval**]: Online Information Services

General Terms

Design, Human Factors

Keywords

content filtering, diversity, Linked Open Data, recommender systems, semantic indexing, semantic recommendation

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

1. INTRODUCTION

Content-based recommender systems observe a user's browsing behaviour and record the interests [1]. By means of natural language processing and machine learning techniques, the user's preferences are extracted and stored in a user profile. The same methods are utilized to obtain suitable content keywords to establish a content profile. Based on previously seen documents, the system attempts to recommend similar content. Therefore, a mathematical representation of the user and content profile is needed. A widely used scheme are TF-IDF (term frequency-inverse document frequency) weights [19]. Computed from the frequency of keywords appearing in a document, these term vectors capture the influence of keywords in a document or preferences in a user profile. The angle between these vectors describes the distance or the closeness of the profiles and is calculated with similarity measures, like the cosine similarity. The recommendation lists of these traditional, keyword-based recommender systems often contain very similar results to those already seen, leading to overspecialization [11] and the "Filter-Bubble"-effect [17]: The user obtains only content according to the stored preferences, other related documents not perfectly matching the stored interests are not displayed. Thus, increasing diversity in recommendations has become an own research area [21, 25, 24, 18, 3, 6, 23], mainly used to improve the recommendation results in news or movie portals.

One field where content recommender systems could enhance daily work is research. Scientists need to be aware of relevant research in their own but also neighboring fields. Increasingly, in addition to literature, the underlying data itself and even data that has not been used in publications are being made publicly available. An important example for such a discipline is biodiversity research, which explores the variety of species and their genetic and characteristic diversity [12]. The morphological and genetic information of an organism, together with the ecological and geographical context, forms a highly diverse structure. Collected and stored in different data formats, the datasets often contain or link to spatial, temporal and environmental data [22]. Many important research questions cannot be answered by working with individual datasets or data collected by one group, but require meta-analysis across a wide range of data. Since the analysis of biodiversity data is quite time-consuming, there is a strong need for personalization and new filtering techniques in this research area. Ordinary search functions in relevant data portals or databases, e.g., the *Global Biodiversity In-*

formation Facility (GBIF)[1] and the *Catalog of Life*,[2] only return data that match the user's query exactly and fail at finding more diverse and semantically related content. Also, user interests are not taken into account in the result list. We believe our semantic-based content recommender system could facilitate the difficult and time-consuming research process in this domain.

Here, we propose a new semantic-based content recommender system that represents the user profile as Linked Open Data (LOD) [9] and incorporates semantic annotations into the recommendation process. Additionally, the search engine is connected to a terminology server and utilizes the provided vocabularies for a recommendation. The result list contains more diverse predictions and includes hierarchical concepts or individuals.

The structure of this paper is as follows: Next, we describe related work. Section 3 presents the architecture of our semantic recommender system and some implementation details. In Section 4, an application scenario is discussed. Finally, conclusions and future work are presented in Section 5.

2. RELATED WORK

The major goal of diversity research in recommender systems is to counteract overspecialization [11] and to recommend related products, articles or documents. More books of an author or different movies of a genre are the classical applications, mainly used in recommender systems based on collaborative filtering methods. In order to enhance the variety in book recommendations, Ziegler et al. [25] enrich user profiles with taxonomical super-topics. The recommendation list generated by this extended profile is merged with a rank in reverse order, called dissimilarity rank. Depending on a certain diversification factor, this merging process supports more or less diverse recommendations. Larger diversification factors lead to more diverse products beyond user interests. Zhang and Hurley [24] favor another mathematical solution and describe the balance between diversity and similarity as a constrained optimization problem. They compute a dissimilarity matrix according to applied criterias, e.g., movie genres, and assign a matching function to find a subset of products that are diverse as well as similar. One hybrid approach by van Setten [21] combines the results of several conventional algorithms, e.g., collaborative and case-based, to improve movie recommendations. Mainly focused on news or social media, approaches using content-based filtering methods try to present different viewpoints on an event to decrease the media bias in news portals [18, 3] or to facilitate the filtering of comments [6, 23].

Apart from Ziegler et al., none of the presented approaches have considered semantic technologies. However, utilizing ontologies and storing user or document profiles in triple stores represents a large potential for diversity research in recommender systems. Frasincar et al. [7] define semantically enhanced recommenders as systems with an underlying knowledge base. This can either be linguistic-based [8], where only linguistic relations (e.g., synonymy, hypernomy, meronymy, antonymy) are considered, or ontology-based. In the latter case, the content and the user profile are represented with concepts of an ontology. This has the advantage that several types of relations can be taken into account. For instance, for a user interested in "geology", the profile contains the concept "geology" that also permits the recommendation of inferred concepts, e.g., "fossil". The idea of recommending related concepts was first introduced by Middelton et al. [15]. They developed *Quickstep*, a recommender system for research papers with ontological terms in the user profile and for paper categories. The ontology only considers is-a relationships and omits other relation types (e.g., part-of). Another simple hierarchical approach from Shoval et al. [13] calculates the distance among concepts in a profile hierarchy. They distinguish between perfect, close and weak match. When the concept appears in both a user's and document's profile, it is called a perfect match. In a close match, the concept emerges only in one of the profiles and a child or parent concept appears in the other. The largest distance is called a weak match, where only one of the profiles contains a grandchild or grandparent concept. Finally, a weighted sum over all matching categories leads to the recommendation list. This ontological filtering method was integrated into the news recommender system *epaper*. Another semantically enhanced recommender system is *Athena* [10]. The underlying ontology is used to explore the semantic neighborhood in the news domain. The authors compared several ontology-based similarity measures with the traditional TF-IDF approach. However, this system lacks of a connection to a search engine that allows to query large datasets.

All presented systems use manually established vocabularies with a limited number of classes. None of them utilize a generic user profile to store the preferences in a semantic format (RDF/XML or OWL). The FOAF (Friend Of A Friend) project[3] provides a vocabulary for describing and connecting people, e.g., demographic information (name, address, age) or interests. As one of the first, in 2006 Celma [2] leveraged FOAF in his music recommender system to store users' preferences. Our approach goes beyond the FOAF interests, by incorporating another generic user model vocabulary, the Intelleo User Modelling Ontology (IUMO).[4] Besides user interests, IUMO offers elements to store learning goals, competences and recommendation preferences. This allows to adapt the results to a user's previous knowledge or to recommend only documents for a specific task.

3. DESIGN AND IMPLEMENTATION

In this section, we describe the architecture and some implementation details of our semantic-based recommender system (Figure 1). The user model component, described in Section 3.1, contains all user information. The source files, described in Section 3.2, are analyzed with GATE [5], as described in Section 3.3. Additionally, GATE is connected with a terminology server (Section 3.2) to annotate documents with concepts from the provided biodiversity vocabularies. In Section 3.4, we explain how the annotated documents are indexed with GATE Mímir [4]. The final recommendation list is generated in the recommender component (Section 3.5).

3.1 User profile

The user interests are stored in an RDF/XML format utilizing the FOAF vocabulary for general user information. In

[1]GBIF, http://www.gbif.org
[2]Catalog of Life, http://www.catalogueoflife.org/col/search/all/
[3]FOAF, http://xmlns.com/foaf/spec/
[4]IUMO, http://intelleo.eu/ontologies/user-model/spec/

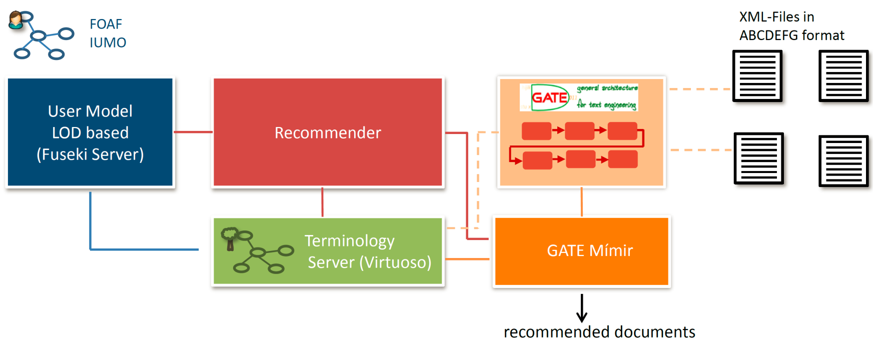

FOAF
IUMO
User Model LOD based (Fuseki Server)
Recommender
GATE general architecture for text engineering
XML-Files in ABCDEFG format
Terminology Server (Virtuoso)
GATE Mímir
recommended documents

Figure 1: The architecture of our semantic content recommender system

order to improve the recommendations regarding a user's previous knowledge and to distinguish between learning goals, interests and recommendation preferences, we incorporate the Intelleo User Modelling Ontology for an extended profile description. Recommendation preferences will contain settings in respect of visualization, e.g., highlighting of interests, and recommender control options, e.g., keyword-search or more diverse results. Another adjustment will adapt the result set according to a user's previous knowledge. In order to enhance the comprehensibility for a beginner, the system could provide synonyms; and for an expert the recommender could include more specific documents.

The interests are stored in form of links to LOD resources. For instance, in our example profile in Listing 1, a user is interested in "biotic mesoscopic physical object", which is a concept from the ENVO[5] ontology. Note that the interest entry in the RDF file does not contain the textual description, but the link to the concept in the ontology, i.e., `http://purl.obolibrary.org/obo/ENVO_01000009`. Currently, we only support explicit user modelling. Thus, the user information has to be added manually to the RDF/XML file. Later, we intend to develop a user profiling component, which gathers a user's interests automatically. The profile is accessible via an Apache Fuseki[6] server.

Listing 1: User profile with interests stored as Linked Open Data URIs

```
<rdf:Description rdf:about="http://www.semanticsoftware.info/person
      /felicitasloeffler">
<rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
<foaf:firstName>Felicitas</foaf:firstName>
<foaf:lastName>Loeffler</foaf:lastName>
<foaf:name>Felicitas Loeffler</foaf:name>
<foaf:gender>Female</foaf:gender>
<foaf:workplaceHomepage rdf:resource="http://dbpedia.org/page/
      University_of_Jena"/>
<foaf:organization>Friedrich Schiller University Jena
</foaf:organization>
<foaf:mbox>felicitas.loeffler@uni-jena.de</foaf:mbox>
<um:TopicPreference rdf:resource="http://purl.obolibrary.org/obo/
      ENVO_01000009"/>
</rdf:Description>
```

3.2 Source files and terminology server

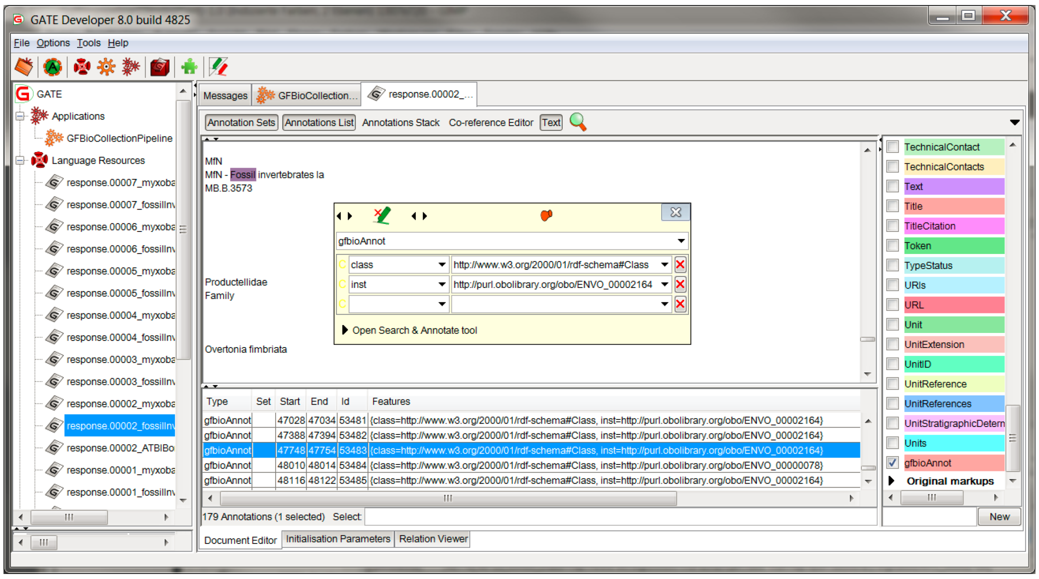
The content provided by our recommender comes from the biodiversity domain. This research area offers a wide range of existing vocabularies. Furthermore, biodiversity is an interdisciplinary field, where the results from several sources have to be linked to gain new knowledge. A recommender system for this domain needs to support scientists by improving this linking process and helping them finding relevant content in an acceptable time.

Researchers in the biodiversity domain are advised to store their datasets together with metadata, describing information about their collected data. A very common metadata format is ABCD.[7] This XML-based standard provides elements for general information (e.g., author, title, address), as well as additional biodiversity related metadata, like information about taxonomy, scientific name, units or gathering. Very often, each taxon needs specific ABCD fields, e.g., fossil datasets include data about the geological era. Therefore, several additional ABCD-related metadata standards have emerged (e.g., ABCDEFG[8], ABCDDNA[9]). One document may contain the metadata of one or more species observations in a textual description. This provides for annotation and indexing for a semantic search. For our prototype, we use the ABCDEFG metadata files provided by the GFBio[10] project; specifically, metadata files from the Museum für Naturkunde (MfN).[11] An example for an ABCDEFG metadata file is presented in Listing 2, containing the core ABCD structure as well as additional information about the geological era. The terminology server supplied by the GFBio project offers access to several biodiversity vocabularies, e.g., ENVO, BEFDATA, TDWGREGION. It also provides a SPARQL endpoint[12] for querying the ontologies.

3.3 Semantic annotation

The source documents are analyzed and annotated according to the vocabularies provided by the terminology server. For this process, we use GATE, an open source framework that offers several standard language engineering components [5]. We developed a custom GATE pipeline (Figure 2) that analyzes the documents: First, the documents are split into tokens and sentences, using the existing NLP components included in the GATE distribution. Afterwards, an 'Annotation Set Transfer' processing resource adds the original

[5]ENVO, `http://purl.obolibrary.org/obo/envo.owl`
[6]Apache Fuseki, `http://jena.apache.org/documentation/serving_data/`
[7]ABCD, `http://www.tdwg.org/standards/115/`
[8]ABCDEFG, `http://www.geocase.eu/efg`
[9]ABCDDNA, `http://www.tdwg.org/standards/640/`
[10]GFBio, `http://www.gfbio.org`
[11]MfN, `http://www.naturkundemuseum-berlin.de/`
[12]GFBio terminology server, `http://terminologies.gfbio.org/sparql/`



MfN
MfN - Fossil invertebrates Ia
MB.B.3573

Productellidae
Family

Overtonia fimbriata

gfbioAnnot

class	http://www.w3.org/2000/01/rdf-schema#Class
inst	http://purl.obolibrary.org/obo/ENVO_00002164

Type	Set	Start	End	Id	Features
gfbioAnnot		47028	47034	53481	{class=http://www.w3.org/2000/01/rdf-schema#Class, inst=http://purl.obolibrary.org/obo/ENVO_00002164}
gfbioAnnot		47388	47394	53482	{class=http://www.w3.org/2000/01/rdf-schema#Class, inst=http://purl.obolibrary.org/obo/ENVO_00002164}
gfbioAnnot		47748	47754	53483	{class=http://www.w3.org/2000/01/rdf-schema#Class, inst=http://purl.obolibrary.org/obo/ENVO_00002164}
gfbioAnnot		48010	48014	53484	{class=http://www.w3.org/2000/01/rdf-schema#Class, inst=http://purl.obolibrary.org/obo/ENVO_00000078}
gfbioAnnot		48116	48122	53485	{class=http://www.w3.org/2000/01/rdf-schema#Class, inst=http://purl.obolibrary.org/obo/ENVO_00002164}

179 Annotations (1 selected) Select:

Figure 2: The GFBio pipeline in GATE presenting the GFBio annotations

markups of the ABCDEFG files to the annotation set, e.g., `abcd:HigherTaxon`. The following ontology-aware 'Large KB Gazetteer' is connected to the terminology server. For each document, all occurring ontology classes are added as specific "gfbioAnnot" annotations that have both instance (link to the concrete source document) and class URI. At the end, a 'GATE Mímir Processing Resource' submits the annotated documents to the semantic search engine.

3.4 Semantic indexing

For semantic indexing, we are using GATE Mímir:[13] "Mímir is a multi-paradigm information management index and repository which can be used to index and search over text, annotations, semantic schemas (ontologies), and semantic metadata (instance data)" [4]. Besides ordinary keyword-based search, Mímir incorporates the previously generated semantic annotations from GATE to the index. Additionally, it can be connected to the terminology server, allowing queries over the ontologies. All index relevant annotations and the connection to the terminology server are specified in an index template.

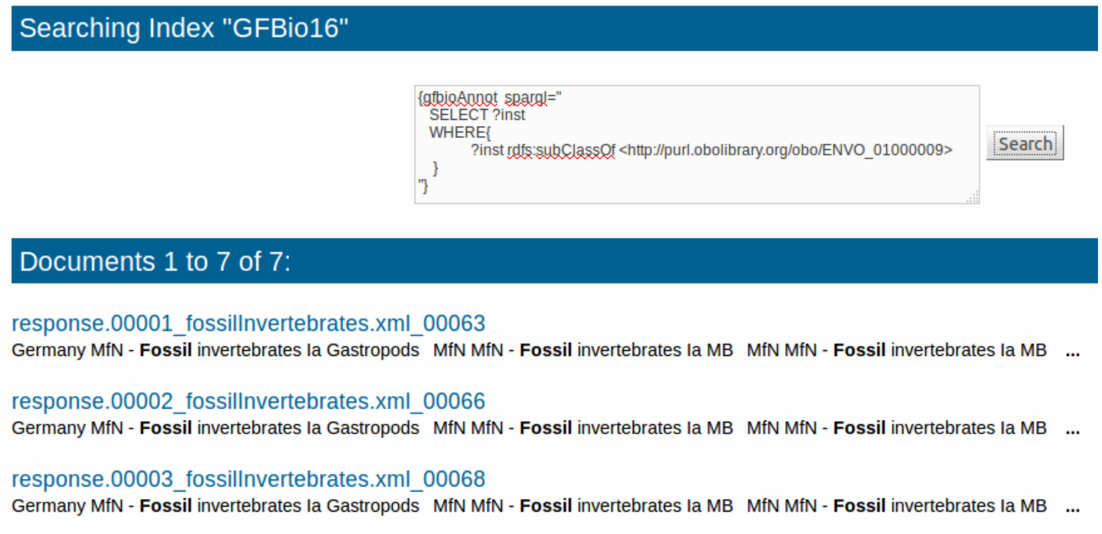
3.5 Content recommender

The Java-based content recommender sends a SPARQL query to the Fuseki Server and obtains the interests and preferred recommendation techniques from the user profile as a list of (LOD) URIs. This list is utilized for a second SPARQL query to the Mímir server. Presently, this query asks only for child nodes (Figure 3). The result set contains ABCDEFG metadata files related to a user's interests. We intend to experiment with further semantic relations in the future, e.g., object properties. Assuming that a specific fossil used to live in rocks, it might be interesting to know if other species, living in this geological era, occured in rocks. Another filtering method would be to use parent or grandparent nodes from the vocabularies to broaden the search. We will provide control options and feedback mechanisms to support the user in steering the recommendation process actively. The recommender component is still under development and has not been added to the implementation yet.

Listing 2: Excerpt from a biodiversity metadata file in ABCDEFG format [20]

```
<abcd:DataSets xmlns:abcd="http://www.tdwg.org/schemas/abcd/2.06"
     xmlns:efg="http://www.synthesys.info/ABCDEFG/1.0">
<abcd:DataSet>
<abcd:Metadata>
<abcd:Description><abcd:Representation language="en">
<abcd:Title>MfN − Fossil invertebrates</abcd:Title>
<abcd:Details>Gastropods, bivalves, brachiopods, sponges</abcd:Details>
    </abcd:Representation></abcd:Description>
<abcd:Scope><abcd:TaxonomicTerms>
<abcd:TaxonomicTerm>Gastropods, Bivalves, Brachiopods, Sponges</
    abcd:TaxonomicTerm>
</abcd:TaxonomicTerms></abcd:Scope>
</abcd:Metadata>
<abcd:Units><abcd:Unit>
<abcd:SourceInstitutionID>MfN</abcd:SourceInstitutionID>
<abcd:SourceID>MfN − Fossil invertebrates Ia</abcd:SourceID>
<abcd:UnitID>MB.Ga.3895</abcd:UnitID>
<abcd:Identifications><abcd:Identification>
<abcd:Result><abcd:TaxonIdentified>
<abcd:HigherTaxa><abcd:HigherTaxon>
<abcd:HigherTaxonName>Euomphaloidea</abcd:HigherTaxonName>
<abcd:HigherTaxonRank>Family</abcd:HigherTaxonRank>
</abcd:HigherTaxon></abcd:HigherTaxa>
<abcd:ScientificName>
<abcd:FullScientificNameString>Euomphalus sp.</
    abcd:FullScientificNameString>
</abcd:ScientificName>
</abcd:TaxonIdentified></abcd:Result>
</abcd:Identification></abcd:Identifications>
<abcd:UnitExtension>
<efg:EarthScienceSpecimen><efg:UnitStratigraphicDetermination>
<efg:ChronostratigraphicAttributions>
<efg:ChronostratigraphicAttribution>
<efg:ChronoStratigraphicDivision>System</
      efg:ChronoStratigraphicDivision>
<efg:ChronostratigraphicName>Triassic</efg:ChronostratigraphicName>
</efg:ChronostratigraphicAttribution></
    efg:ChronostratigraphicAttributions>
</efg:UnitStratigraphicDetermination></efg:EarthScienceSpecimen>
</abcd:UnitExtension>
</abcd:Unit></abcd:Units></abcd:DataSet></abcd:DataSets>
```

[13]GATE Mímir, https://gate.ac.uk/mimir/


Searching Index "GFBio16"

```
{gfbioAnnot sparql="
  SELECT ?inst
  WHERE{
        ?inst rdfs:subClassOf <http://purl.obolibrary.org/obo/ENVO_01000009>
  }
"}
```

Documents 1 to 7 of 7:

response.00001_fossilInvertebrates.xml_00063
Germany MfN - **Fossil** invertebrates Ia Gastropods MfN MfN - **Fossil** invertebrates Ia MB MfN MfN - **Fossil** invertebrates Ia MB ...

response.00002_fossilInvertebrates.xml_00066
Germany MfN - **Fossil** invertebrates Ia Gastropods MfN MfN - **Fossil** invertebrates Ia MB MfN MfN - **Fossil** invertebrates Ia MB ...

response.00003_fossilInvertebrates.xml_00068
Germany MfN - **Fossil** invertebrates Ia Gastropods MfN MfN - **Fossil** invertebrates Ia MB MfN MfN - **Fossil** invertebrates Ia MB ...

Figure 3: A search for "biotic mesoscopic physical object" returning documents about fossils (child concept)

4. APPLICATION

The semantic content recommender system allows the recommendation of more specific and diverse ABCDEFG metadata files with respect to the stored user interests. Listing 3 shows the query to obtain the interests from a user profile, introduced in Listing 1. The result contains a list of (LOD) URIs to concepts in an ontology.

Listing 3: SPARQL query to retrieve user interests

```
SELECT ?label ?interest  ?syn
WHERE
{
      ?s foaf:firstName  "Felicitas" .
      ?s um:TopicPreference ?interest  .
      ?interest  rdfs:label   ?label  .
      ?interest  oboInOwl:hasRelatedSynonym ?syn
}
```

In this example, the user would like to obtain biodiversity datasets about a "biotic mesoscopic physical object", which is the textual description of http://purl.obolibrary.org/obo/ENVO_01000009. This technical term might be incomprehensible for a beginner, e.g., a student, who would prefer a description like "organic material feature". Thus, for a later adjustment of the result according to a user's previous knowledge, the system additionally returns synonyms.

The returned interest (LOD) URI is utilized for a second query to the search engine (Figure 3). The connection to the terminology server allows Mímir to search within the ENVO ontology (Figure 4) and to include related child concepts as well as their children and individuals. Since there is no metadata file containing the exact term "biotic mesoscopic physical object", a simple keyword-based search would fail. However, Mímir can retrieve more specific information than stored in the user profile and is returning biodiversity metadata files about "fossil". That ontology class is a child node of "biotic mesoscopic physical object" and represents a semantic relation. Due to a high similarity regarding the content of the metadata files, the result set in Figure 3 contains only documents which closely resemble each other.

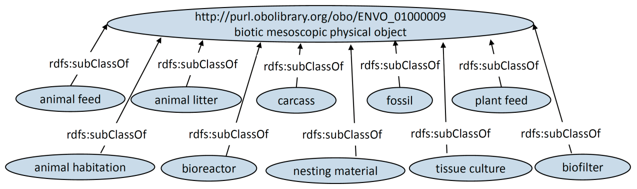


Figure 4: An excerpt from the ENVO ontology

5. CONCLUSIONS

We introduced our new semantically enhanced content recommender system for the biodiversity domain. Its main benefit lays in the connection to a search engine supporting integrated textual, linguistic and ontological queries. We are using existing vocabularies from the terminology server of the GFBio project. The recommendation list contains not only classical keyword-based results, but documents including semantically related concepts.

In future work, we intend to integrate semantic-based recommender algorithms to obtain further diverse results and to support the interdisciplinary linking process in biodiversity research. We will set up an experiment to evaluate the algorithms in large datasets with the established classification metrics *Precision* and *Recall* [14]. Additionally, we would like to extend the recommender component with control options for the user [1]. Integrated into a portal, the result list should be adapted according to a user's recommendation settings or adjusted to previous knowledge. These control functions allow the user to actively steer the recommendation process. We are planning to utilize the new layered evaluation approach for interactive adaptive systems from Paramythis, Weibelzahl and Masthoff [16]. Since adaptive systems present different results to each user, ordinary evaluation metrics are not appropriate. Thus, accuracy, validity, usability, scrutability and transparency will be assessed in several layers, e.g., the collection of input data and their interpretation or the decision upon the adaptation strategy. This should lead to an improved consideration of adaptivity in the evaluation process.

6. ACKNOWLEDGMENTS

This work was supported by DAAD (German Academic Exchange Service)[14] through the PPP Canada program and by DFG (German Research Foundation)[15] within the GFBio project.

7. REFERENCES

[1] F. Bakalov, M.-J. Meurs, B. König-Ries, B. Sateli, R. Witte, G. Butler, and A. Tsang. An approach to controlling user models and personalization effects in recommender systems. In *Proceedings of the 2013 international conference on Intelligent User Interfaces*, IUI '13, pages 49–56, New York, NY, USA, 2013. ACM.

[2] Ò. Celma. FOAFing the music: Bridging the semantic gap in music recommendation. In *Proceedings of 5th International Semantic Web Conference*, pages 927–934, Athens, GA, USA, 2006.

[3] S. Chhabra and P. Resnick. Cubethat: News article recommender. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 295–296, New York, NY, USA, 2012. ACM.

[4] H. Cunningham, V. Tablan, I. Roberts, M. Greenwood, and N. Aswani. Information extraction and semantic annotation for multi-paradigm information management. In M. Lupu, K. Mayer, J. Tait, and A. J. Trippe, editors, *Current Challenges in Patent Information Retrieval*, volume 29 of *The Information Retrieval Series*, pages 307–327. Springer Berlin Heidelberg, 2011.

[5] H. Cunningham et al. *Text Processing with GATE (Version 6)*. University of Sheffield, Dept. of Computer Science, 2011.

[6] S. Faridani, E. Bitton, K. Ryokai, and K. Goldberg. Opinion space: A scalable tool for browsing online comments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1175–1184, New York, NY, USA, 2010. ACM.

[7] F. Frasincar, W. IJntema, F. Goossen, and F. Hogenboom. A semantic approach for news recommendation. *Business Intelligence Applications and the Web: Models, Systems and Technologies, IGI Global*, pages 102–121, 2011.

[8] F. Getahun, J. Tekli, R. Chbeir, M. Viviani, and K. Yétongnon. Relating RSS News/Items. In M. Gaedke, M. Grossniklaus, and O. Díaz, editors, *ICWE*, volume 5648 of *Lecture Notes in Computer Science*, pages 442–452. Springer, 2009.

[9] T. Health and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, 2011.

[10] W. IJntema, F. Goossen, F. Frasincar, and F. Hogenboom. Ontology-based news recommendation. In *Proceedings of the 2010 EDBT/ICDT Workshops*, EDBT '10, pages 16:1–16:6, New York, NY, USA, 2010. ACM.

[11] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[12] M. Loreau. *Excellence in ecology*. International Ecology Institute, Oldendorf, Germany, 2010.

[13] V. Maidel, P. Shoval, B. Shapira, and M. Taieb-Maimon. Ontological content-based filtering for personalised newspapers: A method and its evaluation. *Online Information Review*, 34 Issue 5:729–756, 2010.

[14] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[15] S. E. Middleton, N. R. Shadbolt, and D. C. D. Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, Jan. 2004.

[16] A. Paramythis, S. Weibelzahl, and J. Masthoff. Layered evaluation of interactive adaptive systems: Framework and formative methods. *User Modeling and User-Adapted Interaction*, 20(5):383–453, Dec. 2010.

[17] E. Pariser. *The Filter Bubble - What the internet is hiding from you*. Viking, 2011.

[18] S. Park, S. Kang, S. Chung, and J. Song. Newscube: delivering multiple aspects of news to mitigate media bias. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 443–452, New York, NY, USA, 2009. ACM.

[19] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

[20] Museum für Naturkunde Berlin. Fossil invertebrates, UnitID:MB.Ga.3895. `http://coll.mfn-berlin.de/u/MB_Ga_3895.html`.

[21] M. van Setten. *Supporting people in finding information: hybrid recommender systems and goal-based structuring*. PhD thesis, Telematica Instituut, University of Twente, The Netherlands, 2005.

[22] R. Walls, J. Deck, R. Guralnick, S. Baskauf, R. Beaman, and et al. Semantics in Support of Biodiversity Knowledge Discovery: An Introduction to the Biological Collections Ontology and Related Ontologies. *PLoS ONE 9(3): e89606*, 2014.

[23] D. Wong, S. Faridani, E. Bitton, B. Hartmann, and K. Goldberg. The diversity donut: enabling participant control over the diversity of recommended responses. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 1471–1476, New York, NY, USA, 2011. ACM.

[24] M. Zhang and N. Hurley. Avoiding monotony: Improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 123–130, New York, NY, USA, 2008. ACM.

[25] C.-N. Ziegler, G. Lausen, and L. Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 406–415, New York, NY, USA, 2004. ACM.

[14]DAAD, `https://www.daad.de/de/`
[15]DFG, `http://www.dfg.de`

Exploring Graph Partitioning for Shortest Path Queries on Road Networks

Theodoros Chondrogiannis
Free University of Bozen-Bolzano
tchond@inf.unibz.it

Johann Gamper
Free University of Bozen-Bolzano
gamper@inf.unibz.it

ABSTRACT

Computing the shortest path between two locations in a road network is an important problem that has found numerous applications. The classic solution for the problem is *Dijkstra's* algorithm [1]. Although simple and elegant, the algorithm has proven to be inefficient for very large road networks. To address this deficiency of Dijkstra's algorithm, a plethora of techniques that introduce some preprocessing to reduce the query time have been proposed. In this paper, we propose *Partition-based Shortcuts* (PbS), a technique based on graph-partitioning which offers fast query processing and supports efficient edge weight updates. We present a shortcut computation scheme, which exploits the traits of a graph partition. We also present a modified version of the bidirectional search [2], which uses the precomputed shortcuts to efficiently answer shortest path queries. Moreover, we introduce the Corridor Matrix (CM), a partition-based structure which is exploited to reduce the search space during the processing of shortest path queries when the source and the target point are close. Finally, we evaluate the performance of our modified algorithm in terms of preprocessing cost and query runtime for various graph partitioning configurations.

Keywords

Shortest path, road networks, graph partitioning

1. INTRODUCTION

Computing the shortest path between two locations in a road network is a fundamental problem and has found numerous applications. The problem can be formally defined as follows. Let $G(V, E)$ be a directed weighted graph with vertices V and edges E. For each edge $e \in E$, a weight $l(e)$ is assigned, which usually represents the length of e or the time required to cross e. A path p between two vertices $s, t \in V$ is a sequence of connected edges, $p(s,t) = \langle (s, v_1), (v_1, v_2), \ldots, (v_k, v_t) \rangle$ where $(v_k, v_{k+1}) \in E$, that connects s and t. The shortest path between two vertices s and t is the path $p(s,t)$ that has the shortest distance among all paths that connect s and t.

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26 GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

The classic solution for the shortest path problem is Dijkstra's algorithm [1]. Given a source s and a destination t in a road network G, Dijkstra's algorithm traverses the vertices in G in ascending order of their distances to s. However, Dijkstra's algorithm comes with a major shortcoming. When the distance between the source and the target vertex is high, the algorithm has to expand a very large subset of the vertices in the graph. To address this shortcoming, several techniques have been proposed over the last few decades [3]. Such techniques require a high start-up cost, but in terms of query processing they outperform Dijkstra's algorithm by orders of magnitude.

Although most of the proposed techniques offer fast query processing, the preprocessing is always performed under the assumption that the weights of a road network remain unchanged over time. Moreover, the preprocessing is metric-specific, thus for different metrics the preprocessing needs to be performed for each metric. The recently proposed *Customizable Route Planning* [4] applies preprocessing for various metrics, i.e., distance, time, turn cost and fuel consumption. Such an approach allows a fast computation of shortest path queries using any metric desired by the user, at the cost of some extra space. Moreover, the update cost for the weights is low since the structure is designed such that only a small part of the preprocessed information has to be recomputed. In this paper, our aim is to develop an approach which offers even faster query processing, while keeping the update cost of the preprocessed information low. This is particularly important in dynamic networks, where edge weights might frequently change, e.g., due to traffic jams.

The contributions of this paper can be summarized as follows:

- We present *Partitioned-based Shortcuts* (PbS), a preprocessing method which is based on Customizable Route Planning (CRP), but computes more shortcuts in order to reduce the query processing time.
- We propose the *Corridor Matrix* (CM), a pruning technique which can be used for shortest path queries when the source and the target are very close and the precomputed shortcuts cannot be exploited.
- We run experiments for several different partition configurations and we evaluate our approach in terms of both preprocessing and query processing cost.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe in detail the preprocessing phase of our method. In Section 5, we present a modified version of the bidirectional search algorithm. In Section 6, we show preliminary results of an empirical evaluation. Section 7 concludes the paper and points to future research directions.

2. RELATED WORK

The preprocessing based techniques that have been proposed in order to reduce the time required for processing shortest path queries can be classified into different categories [3]. *Goal-directed techniques* use either heuristics or precomputed information in order to limit the search space by excluding vertices that are not in the direction of the target. For example, A^* [5] search uses the Euclidean distance as a lower bound. *ALT* [6] uses precomputed shortest path distances to a carefully selected set of landmarks and produces the lower bound using the triangle inequality. Some goal-directed techniques exploit graph partitioning in order to prune the search space and speed-up queries. *Precomputed Cluster Distances* (PCD) [7] partitions the graph into k components, computes the distance between all pairs of components and uses the distances between components to compute lower bounds. Arc Flags [8] maintains a vector of k bits for each edge, where the i-th bit is set if the arc lies on a shortest path to some vertex of component i. Otherwise, all edges of component i are pruned by the search algorithm.

Path Coherent techniques take advantage of the fact that shortest paths in road networks are often spatially coherent. To illustrate the concept of spatial coherence, let us consider four locations s, s', t and t' in a road network. If s is close to s' and t is close to t', the shortest path from s to t is likely to share vertices with the shortest path from s' to t'. Spatial coherence methods precompute all shortest paths and use then some data structures to index the paths and answer queries efficiently. For example, Spatially Induced Linkage Cognizance (SILC) [9] use a quad-tree [10] to store the paths. Path-Coherent Pairs Decomposition (PCPD) [11] computes unique path coherent pairs and retrieves any shortest path recursively in almost linear time to the size of the path.

Bounded-hop techniques aim to reduce a shortest path query to a number of look-ups. Transit Node Routing (TNR) [12] is an indexing method that imposes a grid on the road network and recomputes the shortest paths from within each grid cell C to a set of vertices that are deemed important for C (so-called access nodes of C). More approaches are based on the theory of 2-hop labeling [13]. During preprocessing, a label $L(u)$ is computed for each vertex u of the graph such that for any pair u, v of vertices, the distance $dist(u, v)$ can be determined by only looking at the labels $L(u)$ and $L(v)$. A natural special case of this approach is Hub Labeling (HL) [14], in which the label $L(u)$ associated with vertex u consists of a set of vertices (the hubs of u), together with their distances from u.

Finally, *Hierarchical techniques* aim to impose a total order on the nodes as they deem nodes that are crossed by many shortest paths as more important. *Highway Hierarchies* (HH) [15] and its direct descendant *Contraction Hierarchies* (CH) organize the nodes in the road network into a hierarchy based on their relative importance, and create shortcuts among vertices at the same level of the hierarchy. *Arterial Hierarchies* (AH) [16] are inspired by CH, but produce shortcuts by imposing a grid on the graph. AH outperform CH in terms of both asymptotic and practical performance [17]. Some hierarchical approaches exploit graph partition to create shortcuts. *HEPV* [18] and *HiTi* [19] are techniques that pre-computes the distance between any two boundary vertices and create a new overlay graph. By partitioning the overlay graph and repeating the process several times, a hierarchy of partitions is created, which is used to process shortest path queries.

The recent *Customizable Route Planning* (CRP) [4] is the closest work to our own. CRP is able to handle various arbitrary metrics and can also handle dynamic edge weight updates. CRP uses PUNCH [20], a graph partitioning algorithm tailored to road networks. CRP pre-computes distances between boundary vertices in each component and then CRP applies a modified bidirectional search algorithm which expands only the shortcuts and the edges in the source or the target component. The main difference between our approach and CRP is that, instead of computing only shortcuts between border nodes in each component, we compute shortcuts from every node of a component to the border nodes of the same component. The extra shortcuts enable the bidirectional algorithm to start directly from the border nodes, while CRP has to scan the original edges of the source and the target component.

3. PBS PREPROCESSING

The *Partition-based Shortcuts (PbS)* method we propose exploits graph partitioning to produce shortcuts in a preprocessing phase, which during the query phase are used to efficiently compute shortest path queries. The idea is similar to the concept of transit nodes [12]. Every shortest path between two nodes located in different partitions (also termed components) can be expressed as a combination of three smaller shortest paths. Consider the graph in Figure 1 and a query $q(s, t)$, where $s \in C_1$ and $t \in C_5$. The shortest path from s to t can be expressed as $p(s, b_s) + p(b_s, b_t) + p(b_t, t)$, where $b_s \in \{b_1, b_2\}$ and $b_t \in \{b_3, b_4, b_5\}$. Before PbS is able to process shortest path queries, a preprocessing phase is required, which consists of three steps: graph partitioning, in-component shortcut computation and shortcut graph construction.

3.1 Graph Partitioning

The first step in the pre-processing phase is the graph partitioning. Let $G(V, E)$ be a graph with vertices V and edges E. A partition of G is a set $P(G) = \{C_1, \dots, C_k\}$ of connected subgraphs C_i of G, also referred to as *components* of G. For the set $P(G)$, all components must be disjoint, i.e., $C_1 \cap \dots \cap C_k = \emptyset$. Moreover, let $V_1, \dots, V_{|P(G)|}$ be the sets of vertices of each component. The vertex sets of all components must cover the vertex set of the graph, i.e., $V_1 \cup \dots \cup V_{|P(G)|} = V$. We assign a tag to each node of the original graph, which indicates the component the node is located in. The set of *connecting edges*, $E_C \subseteq E$, is the set of all edges in the graph for which the source and target nodes belong to different components, i.e., $(n, n') \in E$ such that $n \in C_i$, $n' \in C_j$ and $C_i \neq C_j$. Finally, we define the *border nodes* of a component C. A node $n \in C$ is a border node of C if there exists a connecting edge $e = (n, n')$ or $e = (n', n)$, i.e., n' is not in C. If $e = (n, n')$, n is called *outgoing border* node of C, whereas if $e = (n', n)$, n is called *incoming border* node of C. The set of all border nodes of a graph is referred to as B. Figure 1 illustrates a graph partitioned into five components. The filled nodes are the border nodes. Note that for ease of exposition we use only undirected graphs in the examples.

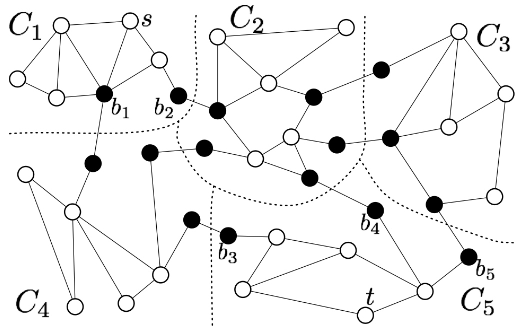


Figure 1: Partitioned graph into five components.

We characterize a graph partition as good if it minimizes the number of connecting edges between the components. However, graph partitioning is an NP-hard problem, thus an optimal solution is out of the question [21]. A popular approach is *multilevel graph partitioning* (MGP), which can be found in many software libraries, such as METIS [22]. Algorithms such as PUNCH [20] and *Spatial Partition Clustering* (SPC) [23] take advantage of road network characteristics in order to provide a more efficient graph partitioning. We use METIS for graph partitioning since it is the most efficient approach out of all available ones [24]. METIS requires only the number of components as an argument in order to perform the partitioning. The number of components influences both the number of the in-component shortcuts and the size of the shortcut graph.

3.2 In-component Shortcuts

The second step of the preprocessing phase is the computation of the in-component shortcuts. For each node n in the original graph, we compute the shortest path from the node to every outgoing border node of the component in which n is located. Then we create outgoing shortcuts which abstract the shortest path from n to each outgoing border node. The incoming shortcuts are computed in a similar fashion. Thus, the total number of in-component shortcuts, S, is

$$S = \sum_{i=1}^{k} N_i \times (|B_{i_{inc}}| + |B_{i_{out}}|),$$

where N_i is the number of nodes in component C_i and $B_{i_{inc}}$, $B_{i_{out}}$ are the incoming and outgoing border nodes of C_i, respectivelly. Figure 2 shows the in-component shortcuts for a node located in component C_2.

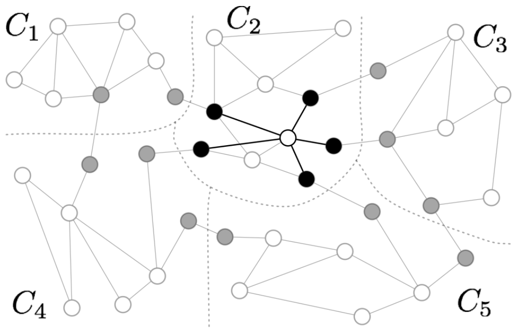


Figure 2: In-component shortcuts for a given node.

For each border node in a component, $b \in C$, we execute Dijkstra's algorithm with b as source and all other nodes (including border nodes) in C as targets. Depending on the type of the source node, the expansion strategy is different. When an incoming border node is the source, forward edges are expanded; vice versa, when an outgoing border node is the source, incoming edges are expanded. This strategy ensures that the maximum number of node expansions is at most twice the number of border nodes of G.

3.3 Shortcut Graph Construction

The third step of the preprocessing phase of our approach is the construction of the shortcut graph. Given a graph G, the shortcut graph of G is a graph $G_{sc}(B, E_{sc})$, where B is the set of border nodes of G and $E_{sc} = E_C \cup S_G$ is the union of the connecting edges, E_C, of G and the shortcuts, S_G, from every incoming border node to every outgoing border node of the same component. Thus, the number of vertices and edges in the shortcut graph is, respectively,

$$|B| = \sum_{i=1}^{k} |B_{i_{inc}} \cup B_{i_{out}}| \text{ and}$$

$$|E_{sc}| = \sum_{i=1}^{k} (|B_{i_{inc}}| \times |B_{i_{out}}|) + E_C.$$

Figure 3 shows the shortcut graph of our running example. Notice that only border nodes are vertices of the shortcut graph. The set of edges consists of connecting edges and the in-component shortcuts between the border nodes of the same component. Note that there is no need for extra computations in order to populate the shortcut graph.

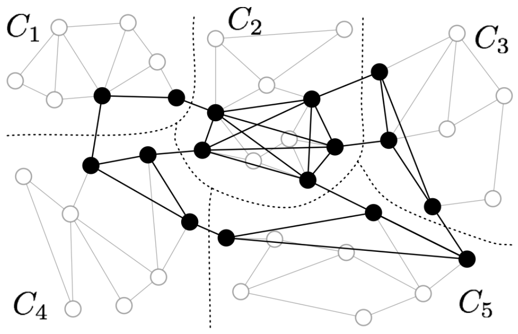
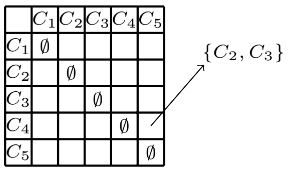


Figure 3: Shortcut Graph illustrated over the original.

4. CORRIDOR MATRIX

In Section 3 we presented how PbS creates shortcuts in order to answer queries when the source and the target points are in different components. However, when the source and the target points of a query are located in the same component, the shortest path may lie entirely inside the component. Therefore, the search algorithm will never reach the border nodes and the shortcuts will not be expanded. In such a case, the common approach is to use bidirectional search to return the shortest path. However, if the components of the partitioned graph are large, the query processing can be quite slow. In order to improve the processing time of such queries, we partition each component again into sub-components, and for each component, we compute its Corridor Matrix (CM). In general, given a partition of a graph G in k components, the *Corridor Matrix* (CM) of G is a $k \times k$ matrix, where each cell $C(i, j)$ of CM contains a list of components that are crossed by some shortest path from a node $s \in C_i$ to a node $t \in C_j$. We call such a list the *corridor* from C_i to C_j. The concept of the CM is similar to Arc-Flags [8], but the CM requires much less space. The space complexity of the CM is $O(k^3)$, where k is the number of components in the partition, while the space complexity of Arc-Flags is $|E| \times k^2$, where $|E|$ is the number of edges in the original graph.



	C_1	C_2	C_3	C_4	C_5
C_1	$\emptyset$				
C_2		$\emptyset$			
C_3			$\emptyset$		
C_4				$\emptyset$	
C_5					$\emptyset$

$\{C_2, C_3\}$

Figure 4: Corridor Matrix example.

To optimize the look-up time in CM, we implemented each component list using a bitmap of length k. Therefore, the space complexity of the CM in the worst case is $O(k^3)$. The actual space occupied by the CM is smaller, since we do not allocate space for bitmaps when the component list is empty. For the computation of the Corridor Matrix, we generate the Shortcut Graph in the same way as described in Section 3.3. To compute the distances between all pairs of vertices, we use the Floyd-Warshall algorithm [25], which is specifically designed to compute the all-pair shortest path distance efficiently. After having computed the distances between the nodes, instead of retrieving each shortest path, we retrieve only the components that are crossed by each path, and we update the CM accordingly.

5. SHORTEST PATH ALGORITHM

In order to process a shortest path query from a source point s to a target point t, we first determine the components of the graph the nodes $s \in C_s$ and $t \in C_t$ are located in. If $C_s = C_t$, we execute a modified bidirectional search from s to t. Note that the shortcuts are not used for processing queries for which the source and target are located in the same component C. Instead, we retrieve the appropriate corridor from the CM of C, which contains a list of sub-components. Then, we apply bidirectional search and prune all nodes that belong to sub-components which are not in the retrieved corridor.

In the case that the points s and t are not located in the same component, we exploit the pre-computed shortcuts. First, we retrieve the lengths of the in-component outgoing shortcuts from s to all the outgoing borders of C_s and the length of the in-component incoming shortcuts from all the incoming borders of C_t to t. Then we apply a many-to-many bidirectional search in the overlay graph from all the outgoing borders of C_s to all the incoming borders of C_t. We use the length of the in-component shortcuts (retrieved in the first step) as initial weights for the source and target nodes of the bidirectional search in the Shortcut Graph. The list of edges consisting the path is a set of connecting edges of the original graph and in-component shortcuts. For each shortcut we retrieve the pre-computed set of the original edges. The cost to retrieve the original path is linear to the size of the path. After the retrieval we replace the shortcuts with the list of edges in the original graph and we return the new edge list, which is the shortest path from s to t in the original graph.

6. PRELIMINARY RESULTS

In this section, we compare our PbS method with CRP, the method our own approach is based on, and CH, a lightweight yet very efficient state-of-the-art approach for shortest path queries in road networks [17]. CRP can handle arbitrary metrics and edge weight updates, while CH is a technique with fast pre-processing and relatively low query processing time. We implemented in Java the basic version of CRP and PbS. The CH algorithm in the experiments is from *Graphhopper Route Planner* [26]. Due to the different implementations of the graph models between ours and CH, we do not measure the runtime. Instead, for preprocessing we count the extra shortcuts created by each algorithm, while for query processing we count the number of expanded nodes.

For the experiments we follow the same evaluation setting as in [17]. We use 5 publicly available datasets [27], four of of which are a part of the US road network, and the smallest one represents the road network of Rome. We present the characteristics of each dataset in Table 1. In order to compare our PbS approach and CRP with CH, we run our experiments over 5 query sets Q_1–$Q5$, which contain 1000 queries each. We make sure that the distance of every query in set Q_i is smaller than the distance of every query in set Q_{i+1}. We also evaluate the CM separately by comparing our CM implementation against Arc Flags and the original bidirectional search for a set of 1000 random queries in the ROME dataset. We use a small dataset in order to simulate in-component query processing.

Name	**Region**	**# Vertices**	**# Edges**
CAL	California/Nevada	1,890,815	4,657,742
FLA	Florida	1,070,376	2,712,798
BAY	SF Bay Area	321,270	800,172
NY	New York City	264,346	733,846
ROME	Center of Rome	3353	8,859

Table 1: Dataset characteristics.

6.1 Preprocessing and Space Overhead

Figures 5 and 6 show a series of measurements for the preprocessing cost of our approach in comparison to CRP and CH over the four largest datasets. Figure 5 shows how many shortcuts are created by each approach. The extra shortcuts can be translated into the space overhead required in order to speed-up shortest path queries. CH uses shortcuts which represent only two edges, while the shortcuts in PbS and CRP are composed of much longer sequences. The difference between the shortcuts produced by CRP and CH is much less. In short, PbS produces about two orders of magnitude more shortcuts than CRP and CH. Moreover, we can observe that the number of shortcuts produced by PbS is getting lower as the number of components is increasing.

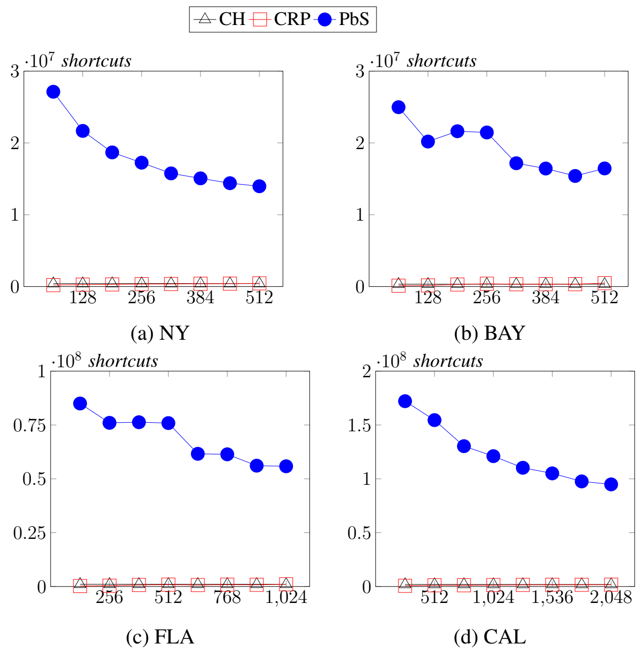


Figure 5: Preprocessing: # of shortcuts vs. # of components.

The same tendency as observed for the number of shortcuts can be observed for the preprocessing time. In Figure 6, we can see that PbS requires much more time than CRP and CH in order to create shortcuts. However, we should also notice that the update cost for CRP and PbS is only a small portion of the preprocessing cost. When an edge weight changes, we need to update only the shortcuts that contains that particular edge. In contrast, for CH the the update cost is the same as the preprocesing cost since a change in a single weight can influence the entire hierarchy.

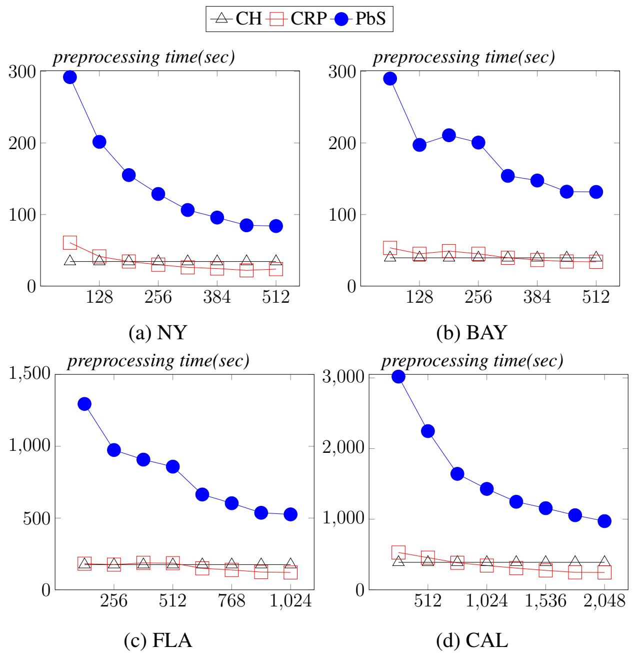


Figure 6: Preprocessing: time vs. # of components.

6.2 Query Processing

Figure 7 shows a series of measurements of the performance of CRP and PbS. We evaluate both techniques for different partitions and various numbers of components. An important observation is the tendency of the performance for CRP and PbS. The performance of CRP gets worse for partitions with many components while the opposite happens for PbS. The reason is that for partitions with few components, PbS manages to process many queries with two look-ups (the case where the source and the target are in adjacent components).

In Figure 8 we compare CH with CRP (we choose the best result) and two configurations of PbS: PbS-BT, which is the configuration that leads to the best performance, and PbS-AVG, which is the average performance of PbS among all configurations. We can see that PbS outperforms CRP in all datasets from Q_1 to Q_5. However, CH is faster in terms of query processing than our PbS approach. CH is more suitable for static networks as the constructed hierarchy of shortcuts enables the shortest path algorithm to expand much fewer nodes.

6.3 In-component Queries

In Figure 9, we compare the performance of our bidirectional algorithm using the proposed CM, the original bidirectional search and the bidirectional algorithm using Arc Flags. We observe that the bidirectional search is the slowest since no pruning is applied. Between Arc Flags and CM, the Arc Flags provide slightly better pruning thus fewer expanded nodes by the bidirectional search. On the other hand, the preprocessing time required to compute the Arc Flags is significantly higher than the time required to compute the CM.

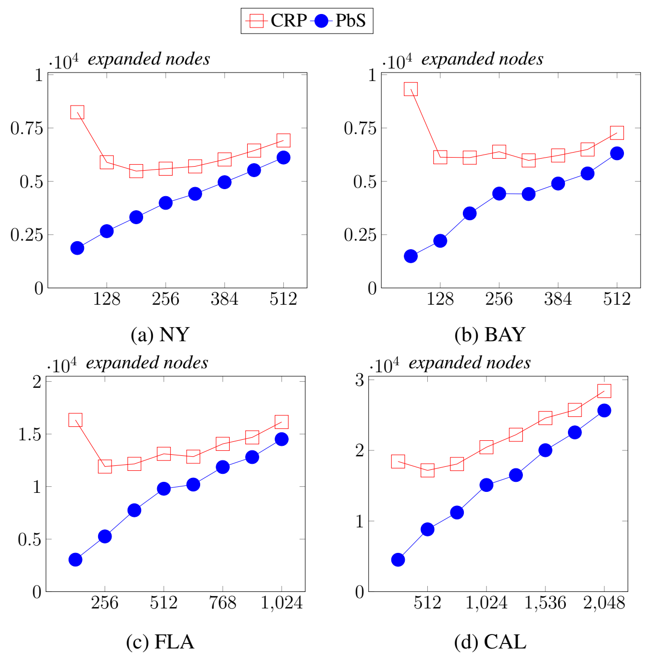


Figure 7: Performance of shortest path queries vs. # of components.

7. CONCLUSION

In this paper we presented PbS, an approach which uses graph partitioning in order to compute shortcuts and speed-up shortest path queries in road networks. Our aim was a solution which supports efficient and incremental updates of edge weights, yet is efficient enough in many real-world applications. In the evaluation, we showed that our PbS approach outperforms CRP. PbS supports edge weight updates as any change in the weight of an edge can influence only shortcuts in a single component. On the other hand, CH is faster than our PbS approach. However, CH cannot handle well edge weight updates as almost the entire hierarchy of shortcuts has to be recomputed every time a single weight changes. For queries where the source and the target are in the same component, we introduced the CM. The efficiency of the CM in query processing approaches the efficiency of Arc Flags, while consuming much less space.

In future work, we plan to extend our approach to support multimodal transportation networks, where the computation has to consider a time schedule, and dynamic and traffic aware networks, where the weights of the edges change over time. We will also improve the preprocessing phase of our approach both in terms of time overhead, by using parallel processing, and space overhead, by using compression techniques or storing some of the precomputed information on the disk.

8. REFERENCES

[1] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[2] I. S. Pohl. *Bi-directional and Heuristic Search in Path Problems*. PhD thesis, Stanford, CA, USA, 1969. AAI7001588.

[3] H. Bast, D. Delling, A. Goldberg, M. Müller, T. Pajor, P. Sanders, D. Wagner, and R Werneck. Route planning in transportation networks. (MSR-TR-2014-4), January 2014.

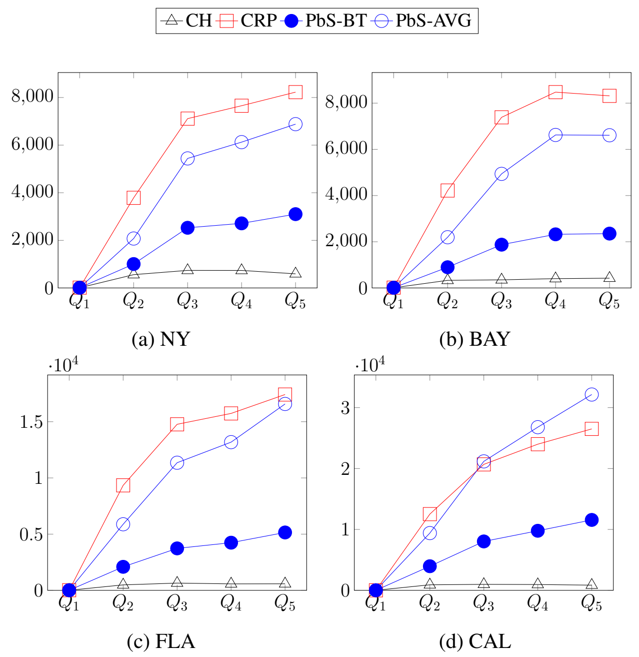


Figure 8: Performance of shortest path queries vs. query sets.

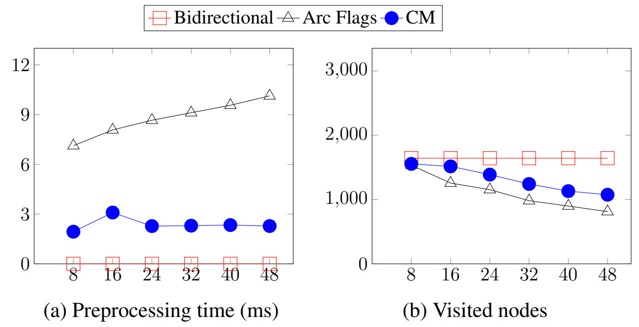


Figure 9: Evaluation of Arc Flags & CM using ROME dataset.

[4] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable route planning. In *Proc. of the 10th Int. Symposium on Experimental Algorithms (SEA)*, pages 376–387, 2011.

[5] P. Hart, N. Nilsson, and B. Raphael. Formal Basis for the Heuristic Determination of Minimum Cost PAths. *IEEE Transactions of Systems Science and Cybernetics*, 4(2):100–107, 1968.

[6] A. V. Goldberg and C. Harrelson. Computing the Shortest Path : A * Search Meets Graph Theory. In *Proc. of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 156–165, 2005.

[7] J. Maue, P. Sanders, and D. Matijevic. Goal-directed shortest-path queries using precomputed cluster distances. *Journal on Experimental Algorithms*, 14:2:3.2–2:3.27, January 2010.

[8] E. Köhler, R. H. Möhring, and H. Schilling. Fast point-to-point shortest path computations with arc-flags. In *Proc. of the 9th DIMACS Implementation Challenge*, 2006.

[9] J. Sankaranarayanan, H. Alborzi, and H. Samet. Efficient query processing on spatial networks. In *Proc. of the 2005 Int. Workshop on Geographic Information Systems (GIS)*, page 200, 2005.

[10] R.A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[11] J. Sankaranarayanan and H. Samet, H. andi Alborzi. Path Oracles for Spatial Networks. In *Proc. of the 35th VLDB Conf.*, pages 1210–1221, 2009.

[12] H. Bast, S. Funke, D Matijevic, P. Sanders, and D. Schultes. In Transit to Constant Time Shortest-Path Queries in Road Networks. In *Proc. of the Workshop on Algorithm Engineering and Experiments*, pages 45–59, 2007.

[13] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. Reachability and distance queries via 2-hop labels. In *Proc. of the 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 937–946, 2002.

[14] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. A hub-based labeling algorithm for shortest paths in road networks. In *Proc. of the 10th Int. Symposium on Experimental Algorithms*, pages 230–241, 2011.

[15] P. Sanders and D. Schultes. Highway Hierarchies Hasten Exact Shortest Path Queries. In *Proc. of the 13th European Conf. on Algorithms (ESA)*, pages 568–579, 2005.

[16] A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, and S. Zhou. Shortest Path and Distance Queries on Road Networks: Towards Bridging Theory and Practice. In *Proc. of the 32nd SIGMOD Conf.*, pages 857–868, 2013.

[17] L. Wu, X. Xiao, D. Deng, G. Cong, and A. D. Zhu. Shortest Path and Distance Queries on Road Networks : An Experimental Evaluation. In *Proc. of the 39th VLDB Conf.*, pages 406–417, 2012.

[18] Y. W. Huang, N. Jing, and E. A. Rundensteiner. Hierarchical path views : A model based on fragmentation and transportation road types. In *Proc. of the 3rd ACM Workshop Geographic Information Systems (GIS),*, 1995.

[19] S. Jung and S. Pramanik. Hiti graph model of topographical roadmaps in navigation systems. In *Proc. of the 12th ICDE Conf.*, pages 76–84, 1996.

[20] D. Delling, A. V. Goldberg, I. Razenshteyn, and R. F. Werneck. Graph Partitioning with Natural Cuts. In *Proc. of the 35th Int. Parallel & Distributed Processing Symposium (IPDPS)*, pages 1135–1146, 2011.

[21] A. E. Feldmann and L/ Foschini. Balanced Partitions of Trees and Applications. In *29th Symp. on Theoretical Aspects of Computer Science*, volume 14, pages 100–111, Paris, France, 2012.

[22] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[23] Y. W. Huang, N. Jing, and E. Rundensteiner. Effective Graph Clustering for Path Queries in Digital Map Databases. In *Proc. of the 5th Int. Conf. on Information and Knowledge Management*, pages 215–222, 1996.

[24] X. Sui, D. Nguyen, M. Burtscher, and K. Pingali. Parallel graph partitioning on multicore architectures. In *Proc. of the 23rd Int. Conf. on Languages and Compilers for Parallel Computing*, pages 246–260, 2011.

[25] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5:345, 1962.

[26] https://graphhopper.com.

[27] http://www.dis.uniroma1.it/challenge9/.

Missing Value Imputation in Time Series using Top-k Case Matching

Kevin Wellenzohn
Free University of Bozen-Bolzano
kevin.wellenzohn@unibz.it

Hannes Mitterer
Free University of Bozen-Bolzano
hannes.mitterer@unibz.it

Johann Gamper
Free University of Bozen-Bolzano
gamper@inf.unibz.it

M. H. Böhlen
University of Zurich
boehlen@ifi.uzh.ch

Mourad Khayati
University of Zurich
mkhayati@ifi.uzh.ch

ABSTRACT

In this paper, we present a simple yet effective algorithm, called the Top-k Case Matching algorithm, for the imputation of missing values in streams of time series data that are similar to each other. The key idea of the algorithm is to look for the k situations in the historical data that are most similar to the current situation and to derive the missing value from the measured values at these k time points. To efficiently identify the top-k most similar historical situations, we adopt Fagin's Threshold Algorithm, yielding an algorithm with sub-linear runtime complexity with high probability, and linear complexity in the worst case (excluding the initial sorting of the data, which is done only once). We provide the results of a first experimental evaluation using real-world meteorological data. Our algorithm achieves a high accuracy and is more accurate and efficient than two more complex state of the art solutions.

Keywords

Time series, imputation of missing values, Threshold Algorithm

1. INTRODUCTION

Time series data is ubiquitous, e.g., in the financial stock market or in meteorology. In many applications time series data is incomplete, that is some values are missing for various reasons, e.g., sensor failures or transmission errors. However, many applications assume complete data, hence need to recover missing values before further data processing is possible.

In this paper, we focus on the imputation of missing values in long streams of meteorological time series data. As a case study, we use real-world meteorological data collected by the *Südtiroler Beratungsring*[1] (*SBR*), which is an organization that provides professional and independent consultancy to the local wine and apple farmers, e.g., to determine the optimal harvesting time or to warn about potential threats, such as apple scab, fire blight, or frost. Especially frost is dangerous as it can destroy the harvest within a few minutes unless the farmers react immediately. The *Südtiroler Beratungsring* operates more than 120 weather stations spread all over South Tyrol, where each of them collects every five minutes up to 20 measurements including temperature, humidity etc. The weather stations frequently suffer outages due to sensor failures or errors in the transmission of the data. However, the *continuous monitoring* of the current weather condition is crucial to immediately warn about imminent threats such as frost and therefore the need arises to recover those missing values as soon as they are detected.

In this paper, we propose an accurate and efficient method to automatically recover missing values. The need for a continuous monitoring of the weather condition at the *SBR* has two important implications for our solution. Firstly, the proposed algorithm has to be efficient enough to complete the imputation before the next set of measurements arrive in a few minutes time. Secondly, the algorithm cannot use future measurements which would facilitate the imputation, since they are not yet available.

The key idea of our Top-k Case Matching algorithm is to seek for the k time points in the historical data when the measured values at a set of reference stations were most similar to the measured values at the current time point (i.e., the time point when a value is missing). The missing value is then derived from the values at the k past time points. While a naïve solution to identify the top-k most similar historical situations would have to scan the entire data set, we adopt Fagin's Threshold Algorithm, which efficiently answers top-k queries by scanning, on average, only a small portion of the data. The runtime complexity of our solution is derived from the Threshold Algorithm and is sub-linear with high probability and linear in the worst case, when all data need to be scanned. We provide the results of a first experimental evaluation using real-world meteorological data from the *SBR*. The results are promising both in terms of efficiency and accuracy. Our algorithm achieves a high accuracy and is more accurate than two state of the art solutions.

The rest of the paper is organized as follows. In Section 2, we review the existing literature about imputation methods for missing values. In Section 3, we introduce the basic notation and a running example. In Section 4, we present our Top-k Case Matching algorithm for the imputation of missing values, followed by the results of an experimental evaluation in Section 5. Section 6 concludes the paper and outlines ideas for future work.

2. RELATED WORK

Khayati et al. [4] present an algorithm, called REBOM, which

[1] http://www.beratungsring.org/

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

recovers blocks of missing values in irregular (with non repeating trends) time series data. The algorithm is based on an iterated truncated matrix decomposition technique. It builds a matrix which stores the time series containing the missing values and its k most correlated time series according to the Pearson correlation coefficient [7]. The missing values are first initialized using a simple interpolation technique, e.g., linear interpolation. Then, the matrix is iteratively decomposed using the truncated *Singular Value Decomposition* (SVD). By multiplying the three matrices obtained from the decomposition, the algorithm is able to accurately approximate the missing values. Due to its quadratic runtime complexity, REBOM is not scalable for long time series data.

Khayati et al. [5] further investigate the use of matrix decomposition techniques for the imputation of missing values. They propose an algorithm with linear space complexity based on the *Centroid Decomposition*, which is an approximation of SVD. Due to the memory-efficient implementation, the algorithm scales to long time series. The imputation follows a similar strategy as the one used in REBOM.

The above techniques are designed to handle missing values in static time series. Therefore, they are not applicable in our scenario, as we have to continuously impute missing values as soon as they appear. A naïve approach to run the algorithms each time a missing value occurs is not feasible due to their relatively high runtime complexity.

There are numerous statistical approaches for the imputation of missing values, including easy ones such as linear or spline interpolation, all the way up to more complex models such as the ARIMA model. The ARIMA model [1] is frequently used for forecasting future values, but can be used for *backcasting* missing values as well, although this is a less common use case. A recent comparison of statistical imputation techniques for meteorological data is presented in [9]. The paper comprises several simple techniques, such as the (weighted) average of concurrent measurements at nearby reference stations, but also computationally more intensive algorithms, such as neural networks.

3. BACKGROUND

Let $\mathbf{S} = \{s_1, \dots, s_n\}$ be a set of time series. Each time series, $s \in \mathbf{S}$, has associated a set of *reference time series* $\mathbf{R}_s$, $\mathbf{R}_s \subseteq \mathbf{S} \setminus \{s\}$. The value of a time series $s \in \mathbf{S}$ at time t is denoted as $s(t)$. A *sliding window* of a time series s is denoted as $s([t_1, t_2])$ and represents all values between t_1 and t_2.

EXAMPLE 1. *Table 1 shows four temperature time series in a time window $w = [1, 7]$, which in our application corresponds to seven timestamps in a range of 30 minutes. s is the base time series from the weather station in* Schlanders, *and $\mathbf{R}_s = \{r_1, r_2, r_3\}$ is the associated set of reference time series containing the stations of Kortsch, Göflan, and Laas, respectively. The temperature value $s(7)$ is missing. Figure 1 visualizes this example graphically.*

The Top-k Case Matching algorithm we propose assumes that the time series data is aligned, which generally is not the case for our data. Each weather station collects roughly every 5 minutes new measurements and transmits them to a central server. Since the stations are not perfectly synchronized, the timestamps of the measurements typically differ, e.g., one station collects measurements at 09:02, 09:07, ..., while another station collects them at 09:04, 09:09, ... Therefore, in a pre-processing step we align the time series data using linear interpolation, which yields measurement values every 5 minutes (e.g., 00:00, 00:05, 00:10, ...). If we observe a gap of more than 10 minutes in the measurements, we assume that the value is missing.

$t \in w$	$s(t)$	$r_1(t)$	$r_2(t)$	$r_3(t)$
1	16.1°	15.0°	15.9°	14.1°
2	15.8°	15.2°	15.7°	13.9°
3	15.9°	15.2°	15.8°	14.1°
4	16.2°	15.0°	15.9°	14.2°
5	16.5°	15.3°	15.7°	14.5°
6	16.1°	15.2°	16.0°	14.1°
7	?	15.0°	16.0°	14.3°

Table 1: Four time series in a window $w = [1, 7]$.

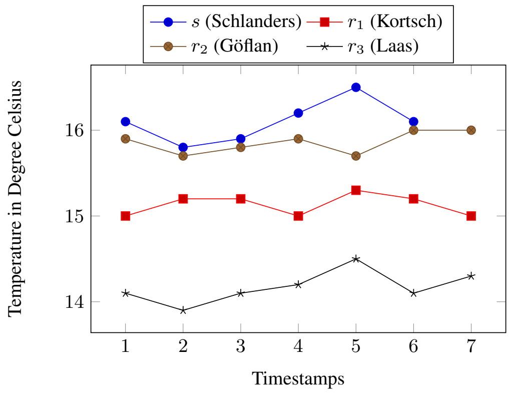


Figure 1: Visualization of the time series data.

For the imputation of missing values we assign to each time series s a set $\mathbf{R}_s$ of reference time series, which are similar to s. The notion of *similarity* between two time series is tricky, though. Intuitively, we want time series to be similar when they have similar values and behave similarly, i.e., values increase and decrease roughly at the same time and by the same amount.

As a simple heuristic for time series similarity, we use the spatial proximity between the stations that record the respective time series. The underlying assumption is that, if the weather stations are nearby (say within a radius of 5 kilometers), the measured values should be similar, too. Based on this assumption, we manually compiled a list of 3–5 reference time series for each time series. This heuristic turned out to work well in most cases, though there are situations where the assumption simply does not hold. One reason for the generally good results is most likely that in our data set the over 100 weather stations cover a relatively small area, and hence the stations are very close to each other.

4. TOP-K CASE MATCHING

Weather phenomena are often repeating, meaning that for example during a hot summer day in 2014 the temperature measured at the various weather stations are about the same as those measured during an equally hot summer day in 2011. We use this observation for the imputation of missing values. Let s be a time series where the current measurement at time θ, $s(\theta)$, is missing. Our assumption on which we base the imputation is as follows: if we find historical situations in the reference time series $\mathbf{R}_s$ such that the past values are very close to the current values at time θ, then also the past measurements in s should be very similar to the missing value $s(\theta)$. Based on this assumption, the algorithm searches for similar climatic situations in historical measurements, thereby leveraging the vast history of weather records collected by the *SBR*.

More formally, given a base time series s with reference time series $\mathbf{R}_s$, we are looking for the k timestamps (i.e., historical situations), $\mathbf{D} = \{t_1, \ldots, t_k\}$, $t_i < \theta$, which minimize the error function

$$\delta(t) = \sum_{r \in \mathbf{R}_s} |r(\theta) - r(t)|.$$

That is, $\delta(t) \leq \delta(t')$ for all $t \in \mathbf{D}$ and $t' \notin \mathbf{D} \cup \{\theta\}$. The error function $\delta(t)$ is the accumulated absolute difference between the current temperature $r(\theta)$ and the temperature at time t, $r(t)$, over all reference time series $r \in \mathbf{R}_s$. Once $\mathbf{D}$ is determined, the missing value is recovered using some aggregation function $g(\{s(t)|\forall t \in \mathbf{D}\})$ over the measured values of the time series s at the timestamps in $\mathbf{D}$. In our experiments we tested the average and the median as aggregation function (cf. Section 5).

EXAMPLE 2. *We show the imputation of the missing value $s(7)$ in Table 1 using as aggregation function g the average. For the imputation, we seek the $k = 2$ most similar historical situations. The two timestamps $\mathbf{D} = \{4, 1\}$ minimize $\delta(t)$ with $\delta(4) = |15.0° - 15.0°| + |16.0° - 15.9°| + |14.3° - 14.2°| = 0.2°$ and $\delta(1) = 0.3°$. The imputation is then simply the average of the base station measurements at time $t = 4$ and $t = 1$, i.e.,$s(7) = avg(16.2°, 16.1°) = \frac{1}{2}(16.2° + 16.1°) = 16.15°$.*

A naïve implementation of this algorithm would have to scan the entire database of historical data to find the k timestamps that minimize $\delta(t)$. This is, however, not scalable for huge time series data, hence a more efficient technique is needed.

4.1 Fagin's Threshold Algorithm

What we are actually trying to do is to answer a top-k query for the k timestamps which minimize $\delta(t)$. There exist efficient algorithms for top-k queries. For example, Fagin's algorithm [2] solves this problem by looking only at a small fraction of the data. Since the first presentation of Fagin's algorithm there were two noteworthy improvements, namely the *Threshold Algorithm* (TA) by Fagin et al. [3] and a probabilistic extension by Theobald et al. [8]. The latter approach speeds up TA by relaxing the requirement to find the exact top-k answers and providing approximations with probabilistic guarantees.

Our Top-k Case Matching algorithm is a variation of TA with slightly different settings. Fagin et al. assume objects with m attributes, a *grade* for each attribute and a *monotone aggregation function* $f : \mathbb{R}^m \mapsto \mathbb{R}$, which aggregates the m grades of an object into an overall grade. The *monotonicity* property is defined as follows.

DEFINITION 1. (Monotonicity) *Let $x_1, \ldots, x_m$ and $x'_1, \ldots, x'_m$ be the m grades for objects X and X', respectively. The aggregation function f is monotone if $f(x_1, \ldots, x_m) \leq f(x'_1, \ldots, x'_m)$ given that $x_i \leq x'_i$ for each $1 \leq i \leq m$.*

The TA finds the k objects that *maximize* the function f. To do so it requires two modes of accessing the data, one being *sorted* and the other *random* access. The sorted access is ensured by maintaining a sorted list $\mathbf{L}^i$ for each attribute m_i, ordered by the grade in descending order. TA keeps a bounded buffer of size k and scans each list $\mathbf{L}^i$ in parallel until the buffer contains k objects and the lowest ranked object in the buffer has an aggregated grade that is greater than or equal to some threshold τ. The threshold τ is computed using the aggregation function f over the grades last seen under the sorted access for each list $\mathbf{L}^i$.

EXAMPLE 3. *Table 2 shows four objects $\{A, B, C, D\}$ and their grade for the two attributes* `interestingness` *and* `popularity`. *Let us assume that $k = 2$ and the aggregation function $f(x_1, x_2) = x_1 + x_2$. Further, assume that the bounded buffer currently contains $\{(C, 18), (A, 16)\}$ and the algorithm has read the data up to the boxes shown in gray. At this point the algorithm computes the threshold using the* `interestingness` *grade for object B and the* `popularity` *grade of object C, yielding $\tau = f(5, 9) = 5 + 9 = 14$. Since the lowest ranked object in the buffer, object A, has an aggregated grade that is greater than τ, we can conclude that C and A are the top-2 objects. Note that the algorithm never read object D, yet it can conclude that D cannot be part of the top-k list.*

interestingness

Object	grade
A	10
C	9
B	5
D	4

popularity

Object	grade
B	10
C	9
D	8
A	6

Table 2: Threshold Algorithm example.

4.2 Adapting the Threshold Algorithm

In order to use the Threshold Algorithm for the imputation of missing values in time series data, we have to adapt it. Instead of looking for the top-k objects that maximize the aggregation function f, we want to find the top-k timestamps that minimize the error function $\delta(t)$ over the reference time series $\mathbf{R}_s$. Similar to TA, we need sorted access to the data. Therefore, for each time series $r \in \mathbf{R}_s$ we define $\mathbf{L}^r$ to be the time series r ordered first by value and then by timestamp in ascending order. Table 3 shows the sorted data for the three reference time series of our running example (ignore the gray boxes and small subscript numbers for the moment).

$\mathbf{L}^{r_1}$

t	$r_1(t)$
1	$15.0°$ $_4$
4	$15.0°$ $_1$
7	$15.0°$
2	$15.2°$
3	$15.2°$
6	$15.2°$
5	$15.3°$

$\mathbf{L}^{r_2}$

t	$r_2(t)$
2	$15.7°$
5	$15.7°$
3	$15.8°$
1	$15.9°$
4	$15.9°$ $_5$
6	$16.0°$ $_2$
7	$16.0°$

$\mathbf{L}^{r_3}$

t	$r_3(t)$
2	$13.9°$
1	$14.1°$
3	$14.1°$
6	$14.1°$
4	$14.2°$ $_3$
7	$14.3°$
5	$14.5°$ $_6$

Table 3: Time series sorted by temperature.

The general idea of our modified TA algorithm is the following. The scan of each sorted lists starts at the current element, i.e., the element with the timestamp $t = \theta$. Instead of scanning the lists $\mathbf{L}^{r_i}$ only in one direction as TA does, we scan each list sequentially in two directions. Hence, as an initialization step, the algorithm places two pointers, pos_r^+ and pos_r^-, at the current value $r(\theta)$ of time series r (the gray boxes in Table 3). During the execution of the algorithm, pointer pos_r^+ is only incremented (i.e., moved down the list), whereas pos_r^- is only decremented (i.e., moved up the list). To maintain the k highest ranking timestamps, the algorithm uses a bounded buffer of size k. A new timestamp t' is added only if the buffer is either not yet full or $\delta(t') < \delta(\underline{t})$, where $\underline{t}$ is the last (i.e., lowest ranking) timestamp in the buffer. In the latter case the timestamp $\underline{t}$ is removed from the buffer.

After this initialization, the algorithm iterates over the lists $\mathbf{L}^r$ in round robin fashion, i.e., once the last list is reached, the algorithm wraps around and continues again with the first list. In each iteration, exactly one list $\mathbf{L}^r$ is processed, and either pointer pos_r^+ or pos_r^- is advanced, depending on which value the two pointers point to has a smaller absolute difference to the current value at time θ, $r(\theta)$. This process grows a neighborhood around the element $r(\theta)$ in each list. Whenever a pointer is advanced by one position, the timestamp t at the new position is processed. At this point, the algorithm needs random access to the values $r(t)$ in each list to compute the error function $\delta(t)$. Time t is added to the bounded buffer using the semantics described above.

The algorithm terminates once the error at the lowest ranking timestamp, $\underline{t}$, among the k timestamps in the buffer is less or equal to the threshold, i.e., $\delta(\underline{t}) \leq \tau$. The threshold τ is defined as $\tau = \sum_{r \in \mathbf{R}_s} |r(\theta) - r(pos_r)|$, where pos_r is either pos_r^+ or pos_r^-, depending on which pointer was advanced last. That is, τ is the sum over all lists $\mathbf{L}^r$ of the absolute differences between $r(\theta)$ and the value under pos_r^+ or pos_r^-.

EXAMPLE 4. *We illustrate the Top-k Case Matching algorithm for $k = 2$ and $\theta = 7$. Table 4 shows the state of the algorithm in each iteration i. The first column shows an iteration counter i, the second the buffer with the k current best timestamps, and the last column the threshold τ. The buffer entries are tuples of the form $(t, \delta(t))$. In iteration $i = 1$, the algorithm moves the pointer to $t = 4$ in list $\mathbf{L}^{r_1}$ and adds $(t = 4, \delta(4) = 0.2°)$ to the buffer. Since $\delta(4) = 0.2° > 0.0° = \tau$, the algorithm continues. The pointer in $\mathbf{L}^{r_2}$ is moved to $t = 6$, and $(6, 0.4°)$ is added to the buffer. In iteration $i = 4$, timestamp 6 is replaced by timestamp 1. Finally, in iteration $i = 6$, the error at timestamp $t = 1$ is smaller or equal to τ, i.e., $\delta(1) = 0.3° \leq \tau_6 = 0.3°$. The algorithm terminates and returns the two timestamps $\mathbf{D} = \{4, 1\}$.*

Iteration i	Buffer	Threshold τ_i
1	$(4, 0.2°)$	0.0°
2	$(4, 0.2°), (6, 0.4°)$	0.0°
3	$(4, 0.2°), (6, 0.4°)$	0.1°
4	$(4, 0.2°), (1, 0.3°)$	0.1°
5	$(4, 0.2°), (1, 0.3°)$	0.2°
6	$(4, 0.2°), (1, 0.3°)$	0.3°

Table 4: Finding the $k = 2$ most similar historical situations.

4.3 Implementation

Algorithm 1 shows the pseudo code of the Top-k Case Matching algorithm. The algorithm has three input parameters: a set of time series $\mathbf{R}_s$, the current timestamp θ, and the parameter k. It returns the top-k most similar timestamps to the current timestamp θ. In line 2 the algorithm initializes the bounded buffer of size k, and in line 4 the pointers pos_r^+ and pos_r^- are initialized for each reference time series $r \in \mathbf{R}_s$. In each iteration of the loop in line 7, the algorithm advances either pos_r^+ or pos_r^- (by calling Algorithm 2) and reads a new timestamp t. The timestamp t is added to the bounded buffer using the semantics described before. In line 15, the algorithm computes the threshold τ. If the buffer contains k timestamps and we have $\delta(\underline{t}) \leq \tau$, the top-$k$ most similar timestamps were found and the algorithm terminates.

Algorithm 2 is responsible for moving the pointers pos_r^+ and pos_r^- for each list $\mathbf{L}^r$. The algorithm uses three utility functions. The first is `next()`, which takes a pointer as input and returns the next position by either incrementing or decrementing, depending on the direction of the pointer. If `next()` reaches the end of a list, it returns NIL. The utility functions `timestamp()` and `value()` return the timestamp and value of a list $\mathbf{L}^r$ at a given position, respectively. There are four cases, which the algorithm has to distinguish:

1. None of the two pointers reached the beginning or end of the list. In this case, the algorithm checks which pointer to advance (line 5). The pointer that is closer to $r(\theta)$ *after* advancing is moved by one position. In case of a tie, we arbitrarily decided to advance pos_r^+.

2. Only pos_r^- reached the beginning of the list: the algorithm increments pos_r^+ (line 11).

3. Only pos_r^+ reached the end of the list: the algorithm decrements pos_r^- (line 13).

4. The two pointers reached the beginning respective end of the list: no pointer is moved.

In the first three cases, the algorithm returns the timestamp that was discovered after advancing the pointer. In the last case, NIL is returned.

At the moment we use an in-memory implementation of the algorithm, which loads the whole data set into main memory. More specifically, we keep two copies of the data in memory: the data sorted by timestamp for fast random access and the data sorted by value and timestamp for fast sorted access.

Note that we did not normalize the raw data using some standard technique like the z-score normalization, as we cannot compute that efficiently for streams of data without increasing the complexity of our algorithm.

Algorithm 1: Top$-k$ Case Matching

```
Data: Reference time series $\mathbf{R}_s$, current time $\theta$, and $k$
Result: $k$ timestamps that minimize $\delta(t)$
1  $L \leftarrow \{\mathbf{L}^r | r \in \mathbf{R}_s\}$
2  buffer $\leftarrow$ boundendBuffer($k$)
3  for $r \in \mathbf{R}_s$ do
4      $pos_r^-, pos_r^+ \leftarrow$ position of $r(\theta)$ in $\mathbf{L}^r$
5  end
6  while $L <> \emptyset$ do
7      for $\mathbf{L}^r \in L$ do
8          $t \leftarrow$ AdvancePointer($\mathbf{L}^r$)
9          if $t =$ NIL then
10             $L \leftarrow L \setminus \{\mathbf{L}^r\}$
11         else
12             if $t \notin$ buffer then
13                 buffer.addWithPriority($t$, $\delta(t)$)
14             end
15             $\tau \leftarrow$ ComputeThreshold($L$)
16             if buffer.size() $= k$
                and buffer.largestError() $\leq \tau$ then
17                 return buffer
18             end
19         end
20     end
21 end
22 return buffer
```

4.4 Proof of Correctness

The correctness of the Top-k Case Matching algorithm follows directly from the correctness of the *Threshold Algorithm*. What remains to be shown, however, is that the aggregation function $\delta(t)$ is monotone.

Algorithm 2: AdvancePointer

```
Data: List $\mathbf{L}^r$ where to advance a pointer
Result: Next timestamp to look at or NIL
1  $pos \leftarrow$ NIL
2  if next($pos_r^+$) <> NIL and next($pos_r^-$) <> NIL then
3      $\Delta^+ \leftarrow |r(\theta) -$ value($\mathbf{L}^r$[next($pos_r^+$)])$|$
4      $\Delta^- \leftarrow |r(\theta) -$ value($\mathbf{L}^r$[next($pos_r^-$)])$|$
5      if $\Delta^+ \leq \Delta^-$ then
6          $pos, pos_r^+ \leftarrow$ next($pos_r^+$)
7      else
8          $pos, pos_r^- \leftarrow$ next($pos_r^-$)
9      end
10 else if next($pos_r^+$) <> NIL and next($pos_r^-$) = NIL then
11     $pos, pos_r^+ \leftarrow$ next($pos_r^+$)
12 else if next($pos_r^+$) = NIL and next($pos_r^-$) <> NIL then
13     $pos, pos_r^- \leftarrow$ next($pos_r^-$)
14 end
15 if $pos$ <> NIL then
16     return timestamp($\mathbf{L}^r[pos]$)
17 else
18     return NIL
19 end
```

THEOREM 4.1. *The aggregation function $\delta(t)$ is a monotonically increasing function.*

PROOF. Let t_1 and t_2 be two timestamps such that $|r(\theta) - r(t_1)| \leq |r(\theta) - r(t_2)|$ for each $r \in \mathbf{R}_s$. Then it trivially follows that $\delta(t_1) \leq \delta(t_2)$ as the aggregation function δ is the sum of $|r(\theta) - r(t_1)|$ over each $r \in \mathbf{R}_s$ and, by definition, each component of $\delta(t_1)$ is less than or equal to the corresponding component in $\delta(t_2)$. $\square$

4.5 Theoretical Bounds

The space and runtime bounds of the algorithm follow directly from the probabilistic guarantees of TA, which has sub-linear cost with high probability and linear cost in the worst case. Note that sorting the raw data to build the lists $\mathbf{L}^r$ is a one-time preprocessing step with complexity $O(n \log n)$. After that the system can insert new measurements efficiently into the sorted lists with logarithmic cost.

5. EXPERIMENTAL EVALUATION

In this section, we present preliminary results of an experimental evaluation of the proposed Top-k Case Matching algorithm. First, we study the impact of parameter k on the Top-k Case Matching and a baseline algorithm. The baseline algorithm, referred to as "Simple Average", imputes the missing value $s(\theta)$ with the average of the values in the reference time series at time θ, i.e., $s(\theta) = \frac{1}{|\mathbf{R}_s|} \sum_{r \in \mathbf{R}_s} r(\theta)$. Second, we compare our solution with two state of the art competitors, REBOM [4] and CD [5].

5.1 Varying k

In this experiment, we study the impact of parameter k on the accuracy and the runtime of our algorithm. We picked five base stations distributed all over South Tyrol, each having two to five reference stations. We simulated a failure of the base station during a time interval, w, of 8 days in the month of April 2013. This amounts to a total of 11452 missing values. We then used the Top-k Case Matching (using both the average and median as aggregation function g) and Simple Average algorithms to impute the missing values. As a measure of accuracy we use the average absolute difference between the real value $s(\theta)$ and the imputed value $s^*(\theta)$, i.e., $\Delta = \frac{1}{|w|} \sum_{\theta \in w} |s(\theta) - s^*(\theta)|$

Figure 2 shows how the accuracy of the algorithms changes with varying k. Interestingly and somewhat unexpectedly, Δ *decreases* as k *increases*. This is somehow contrary to what we expected, since with an increasing k also the error function $\delta(t)$ grows, and therefore less similar historical situations are used for the imputation. However, after a careful analysis of the results it turned out that for low values of k the algorithm is more sensitive to outliers, and due to the often low quality of the raw data the imputation is flawed.

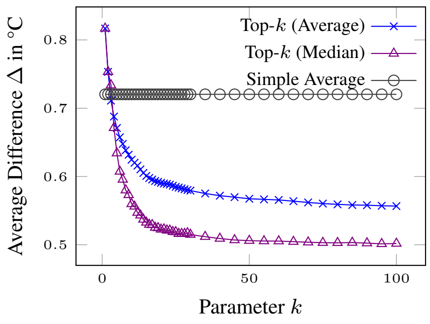


Figure 2: Impact of k on accuracy.

Table 5 shows an example of flawed raw data. The first row is the current situation, and we assume that the value in the gray box is missing and need to be recovered. The search for the $k = 3$ most similar situations using our algorithm yields the three rows at the bottom. Notice that one base station value is 39.9° around midnight of a day in August, which is obviously a very unlikely thing to happen. By increasing k, the impact of such outliers is reduced and hence Δ decreases. Furthermore, using the median as aggregation function reduces the impact of outliers and therefore yields better results than the average.

Timestamp	s	r_1	r_2	r_3
2013-04-16 19:35	18.399°	17.100°	19.293°	18.043°
2012-08-24 01:40	18.276°	17.111°	19.300°	18.017°
2004-09-29 15:50	19.644°	17.114°	19.259°	18.072°
2003-08-02 01:10	39.900°	17.100°	19.365°	18.065°

Table 5: Example of flawed raw data.

Figure 3 shows the runtime, which for the Top-k Case Matching algorithm linearly increases with k. Notice that, although the imputation of missing values for 8 days takes several minutes, the algorithm is fast enough to continuously impute missing values in our application at the *SBR*. The experiment essentially corresponds to a scenario, where in 11452 base stations an error occurs at the same time. With 120 weather stations operated by the *SBR*, the number of missing values at each time is only a tiny fraction of the missing values that we simulated in this experiment.

5.2 Comparison with CD and REBOM

In this experiment, we compare the Top-k Case Matching algorithm with two state-of-the-art algorithms, REBOM [4] and CD [5]. We used four time series, each containing 50.000 measurements, which corresponds roughly to half a year of temperature measurements. We simulated a week of missing values (i.e., 2017 measurements) in one time series and used the other three as reference time series for the imputation.

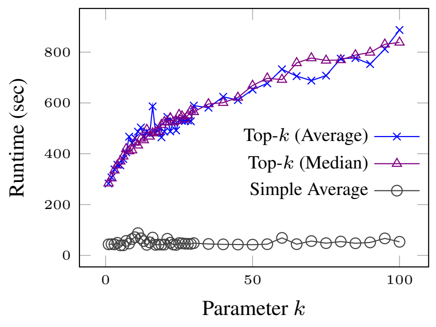


Figure 3: Impact of k on runtime.

The box plot in Figure 4 shows how the imputation error $|s(\theta) - s^*(\theta)|$ is distributed for each of the four algorithms. The left and right line of the box are the first and third quartile, respectively. The line inside the box denotes the median and the left and right whiskers are the 2.5% and 97.5% percentile, which means that the plot incorporates 95% of the values and omits statistical outliers. The experiment clearly shows that the Top-k Case Matching algorithm is able to impute the missing values more accurately than CD and REBOM. Although not visualized, also the maximum observed error for our algorithm is with 2.29° (Average) and 2.21° (Median) considerably lower than 3.71° for CD and 3.6° for REBOM.

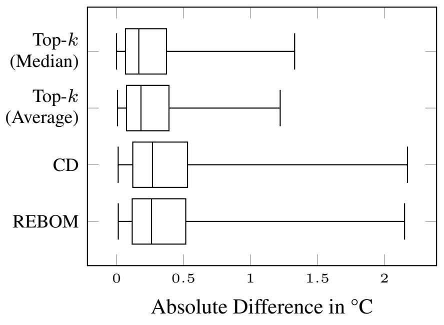


Figure 4: Comparison with REBOM and CD.

In terms of runtime, the Top-k Case Matching algorithm needed 16 seconds for the imputation of the 2017 missing measurements, whereas CD and REBOM needed roughly 10 minutes each. Note, however, that this large difference in run time is also due to the fact that CD and REBOM need to compute the Pearson correlation coefficient which is a time intensive operation.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a simple yet efficient and accurate algorithm, termed Top-k Case Matching, for the imputation of missing values in time series data, where the time series are similar to each other. The basic idea of the algorithm is to look for the k situations in the historical data that are most similar to the current situation and to derive the missing values from the data at these time points. Our Top-k Case Matching algorithm is based on Fagin's Threshold Algorithm. We presented the results of a first experimental evaluation. The Top-k Case Matching algorithm achieves a high accuracy and outperforms two state of the art solutions both in terms of accuracy and runtime.

As next steps we will continue with the evaluation of the algorithm, taking into account also model based techniques such as DynaMMo [6] and other statistical approaches outlined in [9]. We will further study the impact of complex weather phenomena that we observed in our data, such as the foehn. The foehn induces shifting effects in the time series data, as the warm wind causes the temperature to increase rapidly by up to 15° as soon as the foehn reaches another station.

There are several possibilities to further improve the algorithm. First, we would like to explore whether the algorithm can dynamically determine an optimal value for the parameter k, which is currently given by the user. Second, we would like to make the algorithm more robust against outliers. For example, the algorithm could consider only historical situations that occur roughly at the same time of the day. Moreover, we can bend the definition of "current situation" to not only consider the current timestamp, but rather a small window of consecutive timestamps. This should make the ranking more robust against anomalies in the raw data and weather phenomena such as the foehn. Third, right now the similarity between time series is based solely on temperature data. We would like to include the other time series data collected by the weather stations, such as humidity, precipitation, wind, etc. Finally, the algorithm should be able to automatically choose the currently handpicked reference time series based on some similarity measures, such as the Pearson correlation coefficient.

7. ACKNOWLEDGEMENTS

The work has been done as part of the DASA project, which is funded by the Foundation of the Free University of Bozen-Bolzano. We wish to thank our partners at the Südtiroler Beratungsring and the Research Centre for Agriculture and Forestry Laimburg for the good collaboration and helpful domain insights they provided, in particular Armin Hofer, Martin Thalheimer, and Robert Wiedmer.

8. REFERENCES

[1] G. E. P. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[2] R. Fagin. Combining fuzzy information from multiple systems (extended abstract). In *PODS'96*, pages 216–226, New York, NY, USA, 1996. ACM.

[3] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS '01*, pages 102–113, New York, NY, USA, 2001. ACM.

[4] M. Khayati and M. H. Böhlen. REBOM: recovery of blocks of missing values in time series. In *COMAD'12*, pages 44–55, 2012.

[5] M. Khayati, M. H. Böhlen, and J. Gamper. Memory-efficient centroid decomposition for long time series. In *ICDE'14*, pages 100–111, 2014.

[6] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos. Dynammo: Mining and summarization of coevolving sequences with missing values. In *KDD'09*, pages 507–516, New York, NY, USA, 2009. ACM.

[7] A. Mueen, S. Nath, and J. Liu. Fast approximate correlation for massive time-series data. In *SIGMOD'10*, pages 171–182, New York, NY, USA, 2010. ACM.

[8] M. Theobald, G. Weikum, and R. Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB'04*, pages 648–659. VLDB Endowment, 2004.

[9] C. Yozgatligil, S. Aslan, C. Iyigun, and I. Batmaz. Comparison of missing value imputation methods in time series: the case of turkish meteorological data. *Theoretical and Applied Climatology*, 112(1-2):143–167, 2013.

Dominanzproblem bei der Nutzung von Multi-Feature-Ansätzen

Thomas Böttcher
Technical University Cottbus-Senftenberg
Walther-Pauer-Str. 2, 03046 Cottbus
tboettcher@tu-cottbus.de

Ingo Schmitt
Technical University Cottbus-Senftenberg
Walther-Pauer-Str. 2, 03046 Cottbus
schmitt@tu-cottbus.de

ABSTRACT

Ein Vergleich von Objekten anhand unterschiedlicher Eigenschaften liefert auch unterschiedliche Ergebnisse. Zahlreiche Arbeiten haben gezeigt, dass die Verwendung von mehreren Eigenschaften signifikante Verbesserungen im Bereich des Retrievals erzielen kann. Ein großes Problem bei der Verwendung mehrerer Eigenschaften ist jedoch die Vergleichbarkeit der Einzeleigenschaften in Bezug auf die Aggregation. Häufig wird eine Eigenschaft von einer anderen dominiert. Viele Normalisierungsansätze versuchen dieses Problem zu lösen, nutzen aber nur eingeschränkte Informationen. In dieser Arbeit werden wir einen Ansatz vorstellen, der die Messung des Grades der Dominanz erlaubt und somit auch eine Evaluierung verschiedener Normalisierungsansätze.

Keywords

Dominanz, Score-Normalisierung, Aggregation, Feature

1. EINLEITUNG

Im Bereich des Information-Retrievals (IR), Multimedia-Retrievals (MMR), Data-Mining (DM) und vielen anderen Gebieten ist ein Vergleich von Objekten essentiell, z.B. zur Erkennung ähnlicher Objekte bzw. Duplikate oder zur Klassifizierung der untersuchten Objekte. Der Vergleich von Objekten einer Objektmenge O basiert dabei in der Regel auf deren Eigenschaftswerten. Im Bereich des MMR sind Eigenschaften (Features) wie Farben, Kanten oder Texturen häufig genutzte Merkmale. In vielen Fällen genügt es für einen erschöpfenden Vergleich von Objekten nicht, nur eine Eigenschaft zu verwenden. Abbildung 1 zeigt anhand des Beispiels eines Farbhistogramms die Schwächen einer einzelnen Eigenschaft. Obwohl beide Objekte sich deutlich unterscheiden so weisen sie ein sehr ähnliches Farbhistogramm auf.
Statt einer Eigenschaft sollte vielmehr eine geeignete Kombination verschiedener Merkmale genutzt werden, um mittels einer verbesserten Ausdruckskraft [16] genauere Ergebnissen zu erzielen. Der (paarweise) Vergleich von Objekten anhand

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

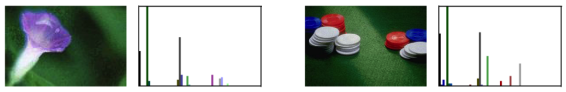


Figure 1: Unterschiedliche Objekte mit sehr hoher Farbähnlichkeit

von Eigenschaften erfolgt mittels eines Distanz- bzw. Ähnlichkeitsmaßes[1]. Bei der Verwendung mehrerer Eigenschaften lassen sich Distanzen mittels einer Aggregationsfunktion verknüpfen und zu einer Gesamtdistanz zusammenfassen. Der Einsatz von unterschiedlichen Distanzmaßen und Aggregationsfunktionen bringt jedoch verschiedene Probleme mit sich:
Verschiedene Distanzmaße erfüllen unterschiedliche algebraische Eigenschaften und nicht alle Distanzmaße sind für spezielle Probleme gleich geeignet. So erfordern Ansätze zu metrischen Indexverfahren oder Algorithmen im Data-Mining die Erfüllung der Dreiecksungleichung. Weitere Probleme können durch die Eigenschaften der Aggregationsfunktion auftreten. So kann diese z.B. die Monotonie oder andere algebraische Eigenschaften der Einzeldistanzmaße zerstören. Diese Probleme sollen jedoch nicht im Fokus dieser Arbeit stehen.
Für einen Ähnlichkeitsvergleich von Objekten anhand mehrerer Merkmale wird erwartet, dass die Einzelmerkmale gleichermaßen das Aggregationsergebnis beeinflussen. Häufig gibt es jedoch ein Ungleichgewicht, welches die Ergebnisse so stark beeinflusst, dass einzelne Merkmale keinen oder nur einen geringen Einfluss besitzen. Fehlen algebraische Eigenschaften oder gibt es eine zu starke Dominanz, so können die Merkmale und dazugehörigen Distanzmaße nicht mehr sinnvoll innerhalb einer geeigneten Merkmalskombination eingesetzt werden. Im Bereich der Bildanalyse werden zudem immer komplexere Eigenschaften aus den Bilddaten extrahiert. Damit wird auch die Berechnung der Distanzen basierend auf diesen Eigenschaften immer spezieller und es kann nicht sichergestellt werden welche algebraische Eigenschaften erfüllt werden. Durch die vermehrte Verwendung von vielen Einzelmerkmalen steigt auch das Risiko der Dominanz eines oder weniger Merkmale.
Kernfokus dieser Arbeit ist dabei die Analyse von Multi-Feature-Aggregationen in Bezug auf die Dominanz einzelner Merkmale. Wir werden zunächst die Dominanz einer Eigen-

[1]Beide lassen sich ineinander überführen [Sch06], im Folgenden gehen wir daher von Distanzmaßen aus.

schaft definieren und zeigen wann sich eine solche Dominanz manifestiert. Anschließend führen wir ein Maß zur Messung des Dominanzgrades ein. Wir werden darüber hinaus zeigen, dass die Ansätze bestehender Normalisierungsverfahren nicht immer ausreichen um das Problem der Dominanz zu lösen. Zusätzlich ermöglicht dieses Maß die Evaluation verschiedener Normalisierungsansätze.
Die Arbeit ist dabei wie folgt aufgebaut. In Kapitel 2 werden noch einmal einige Grundlagen zur Distanzfunktion und zur Aggregation dargelegt. Kapitel 3 beschäftigt sich mit der Definition der Dominanz und zeigt anhand eines Beispiels die Auswirkungen. Weiterhin wird ein neues Maß zur Messung des Dominanzgrades vorgestellt. Kapitel 4 liefert einen Überblick über bestehende Ansätze. Kapitel 5 gibt eine Zusammenfassung und einen Ausblick für zukünftige Arbeiten.

2. GRUNDLAGEN

Das folgende Kapitel definiert die grundlegenden Begriffe und die Notationen, die in dieser Arbeit verwendet werden. Distanzberechnungen auf unterschiedlichen Merkmalen erfordern in der Regel auch den Einsatz unterschiedlicher Distanzmaße. Diese sind in vielen Fällen speziell auf die Eigenschaft selbst optimiert bzw. angepasst. Für eine Distanzberechnung auf mehreren Merkmalen werden dementsprechend auch unterschiedliche Distanzmaße benötigt.
Ein Distanzmaß zwischen zwei Objekten basierend auf einer Eigenschaft p sei als eine Funktion $d : O \times O \mapsto \mathbb{R}_{\geq 0}$ definiert. Ein Distanzwert basierend auf einem Objektvergleich zwischen o_r und o_s über einer einzelnen Eigenschaft p_j wird mit $d^j(o_r, o_s) \in \mathbb{R}_{\geq 0}$ beschrieben. Unterschiedliche Distanzmaße besitzen damit auch unterschiedliche Eigenschaften. Zur Klassifikation der unterschiedlichen Distanzmaße werden folgende vier Eigenschaften genutzt:
Selbstidentität: $\forall o \in O : d(o, o) = 0$, Positivität: $\forall o_r \neq o_s \in O : d(o_r, o_s) > 0$, Symmetrie: $\forall o_r, o_s \in O : d(o_r, o_s) = d(o_s, o_r)$ und Dreiecksungleichung: $\forall o_r, o_s, o_t \in O : d(o_r, o_t) \leq d(o_r, o_s) + d(o_s, o_t)$.
Erfüllt eine Distanzfunktion alle vier Eigenschaften so wird sie als Metrik bezeichnet [11].
Ist der Vergleich zweier Objekte anhand einer einzelnen Eigenschaft nicht mehr ausreichend, um die gewünschte (Un-) Ähnlichkeit für zwei Objekte o_r,$o_s \in O$ zu bestimmen , so ist die Verwendung mehrerer Eigenschaften nötig. Für eine Distanzberechnung mit m Eigenschaften $p = (p_1 \dots p_m)$ werden zunächst die partiellen Distanzen $\delta^j_{rs} = d^j(o_r, o_s)$ bestimmt. Anschließend werden die partiellen Distanzwerte δ^j_{rs} mittels einer Aggregationsfunktion $agg : \mathbb{R}^m_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ zu einer Gesamtdistanz aggregiert. Die Menge aller aggregierten Distanzen (Dreiecksmatrix) für Objektpaar aus O, sei durch $\delta^j = (\delta^j_1, \delta^j_2 \dots, \delta^j_l)$ mit $l = \frac{n^2-n}{2}$ bestimmt. Dieser Ansatz erlaubt eine Bestimmung der Aggregation auf den jeweiligen Einzeldistanzwerten. Die Einzeldistanzfunktionen d^j sind in sich geschlossen und damit optimiert auf die Eigenschaft selbst.

3. DOMINANZPROBLEM

Bisher haben wir das Problem der Dominanz nur kurz eingeführt. Eine detaillierte Motivation und Heranführung an das Problem soll in diesem Kapitel erfolgen. Hierzu werden wir zunächst die Begriffe *Überbewertung* und *Dominanzproblem* einführen. Die Auswirkungen des Dominanzproblem auf das Aggregationsergebnis sollen anschließend durch ein Beispiel erläutert werden. Abschließend werden wir ein Maß definieren, um den Grad der Dominanz messen zu können.

3.1 Problemdefinition

Wie bereits erwähnt ist der Einsatz vieler, unterschiedlicher Eigenschaften (Features) und ihrer teilweise speziellen Distanzmaße nicht trivial und bringt einige Herausforderungen mit sich. Das Problem der Dominanz soll in diesem Unterabschnitt noch einmal genauer definiert werden.

Zunächst definieren wir das Kernproblem bei der Aggregation mehrerer Distanzwerte.

PROBLEM: *Für einen Ähnlichkeitsvergleich von Objekten anhand mehrerer Merkmale sollen die Einzelmerkmale gleichermaßen das Aggregationsergebnis beeinflussen. Dominieren die partiellen Distanzen δ^j_{rs} eines Distanzmaßes d^j das Aggregationsergebnis, so soll diese Dominanz reduziert bzw. beseitigt werden.*
Offen ist an dieser Stelle die Frage, wann eine Dominanz einer Eigenschaft auftritt, wie sich diese auf das Aggregationsergebnis auswirkt und wie der Grad der Dominanz gemessen werden kann.
Das Ergebnis einer Aggregation von Einzeldistanzwerten ist erneut ein Distanzwert. Dieser soll jedoch von allen Einzeldistanzwerten gleichermaßen abhängen. Ist der Wertebereich, der zur Aggregation verwendeten Distanzfunktionen nicht identisch, so kann eine Verfälschung des Aggregationsergebnisses auftreten. Als einfaches Beispiel seien hier zwei Distanzfunktionen d_1 und d_2 genannt, wobei d_1 alle Distanzen auf das Intervall $[0, 1]$ und d_2 alle Distanzen auf $[0, 128]$ abbildet. Betrachtet man nun eine Aggregationsfunktion d_{agg}, die Einzeldistanzen aufsummiert, so zeigt sich, dass d_2 das Aggregationsergebnis erheblich mehr beeinflusst als d_1.
Allgemein werden dann die aggregierten Distanzwerte stärker oder schwächer durch Einzeldistanzwerte einer (zur Aggregation verwendeten) Distanzfunktion beeinflusst als gewünscht. Wir bezeichnen diesen Effekt als eine Überwertung. Der Grad der Überbewertung lässt sich mittels Korrelationsanalyse (z.B. nach Pearson [10] oder Spearman [13]) bestimmen.

DEFINITION 1 (*Überbewertung einer Distanzfunktion*). *Für zwei Distanzfunktionen d^j und d^k, bei der die Distanzwerte δ^j in Abhängigkeit einer Aggregationsfunktion agg das Aggregationsergebnis stärker beeinflussen als δ^k, also die Differenz der Korrelationswerte*
$\rho(\delta^j, \delta^{agg}) - \rho(\delta^k, \delta^{agg}) > \epsilon$ ist, bezeichnen wir d^j als überbewertet gegenüber d^k.

Eine empirische Untersuchung hat gezeigt, dass sich ab einem Wert $\epsilon \geq 0.2$ eine Beeinträchtigung des Aggregationsergebnisses zu Gunsten einer Distanzfunktion zeigt.
Ausgehend von einer Überbewertung definieren wir das Problem der Dominanz.

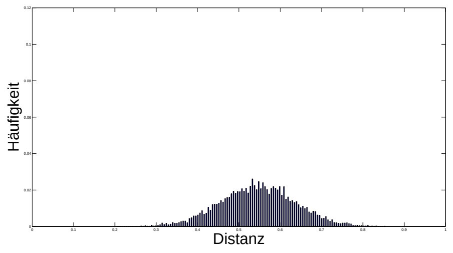
DEFINITION 2 (*Dominanzproblem*). *Ein Dominanzproblem liegt vor, wenn es eine Überbewertung einer Distanzfunktion d^j gegenüber d^k gibt.*

Das Problem einer Überbewertung bei unterschiedlichen Wertebereichen in denen die Distanzen abgebildet werden ist jedoch bereits weitreichend bekannt. In vielen Fällen kommen Normalisierungsverfahren (z.B. im Data-Mining [12] oder in der Biometrie [5]) zum Einsatz. Diese bereiten Distanzen aus verschiedenen Quellen für eine Aggregation vor.

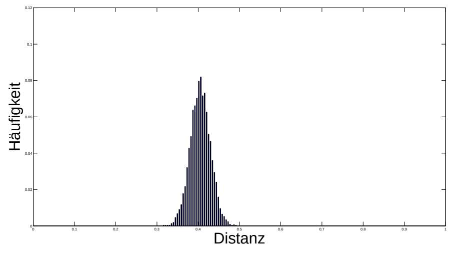
Zur Vermeidung einer Überbewertung werden Distanzen häufig auf ein festes Intervall normalisiert (i.d.R. auf [0,1]). Damit ist zumindest das Problem in unserem vorherigen Beispiel gelöst.
Das Problem der Dominanz tritt jedoch nicht nur bei unterschiedlichen Wertebereichen auf. Auch bei Distanzfunktionen, die alle auf den gleichen Wertebereich normalisiert sind, kann das Dominanzproblem auftreten. Im folgenden Abschnitt soll anhand eines Beispiels dieses Dominanzproblem demonstriert werden.

3.2 Beispiel eines Dominanzproblems

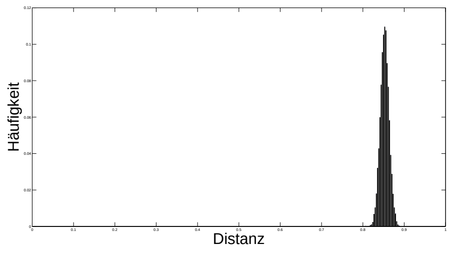
In Abbildung 2 sind drei Distanzverteilungen ν_1, ν_2 und ν_3 aus einer Stichprobe zu den zugehörigen Distanzfunktionen d_1, d_2 sowie d_3 dargestellt. Der Wertebereich der Funktionen sei auf das Intervall [0,1] definiert. Die Werte aus der Stichprobe treten ungeachtet der Normalisierung auf $[0, 1]$ jedoch in unterschiedlichen Intervallen auf. Die Distanzwerte der Stichprobe von ν_1 liegen im Intervall [0.2, 0.9], von ν_2 im Intervall [0.3, 0.5] und in ν_3 im Intervall [0.8, 0.9]. Auch wenn es sich hierbei um simulierte Daten handelt so sind solche Verteilungen im Bereich des MMR häufig anzutreffen.



(a) ν_1



(b) ν_2



(c) ν_3

Figure 2: Distanzverteilung verschiedener Distanzfunktionen (simulierte Daten)

Wir betrachten nun die Distanzfunktionen d_1 und d_2. Bezüglich einer beispielhaften Aggregationsfunktion[2] $agg_{\prod_{d_1,d_2}}(o_r, o_s) = d_1(o_r, o_s) * d_2(o_r, o_s)$ kann nun gezeigt werden, dass d_1 stärker den aggregierten Distanzwert beeinflusst als d_2.
In Abbildung 3 sind zwei verschiedene Rangfolgen aller 10 Distanzwerte zwischen fünf zufälligen Objekten der Verteilungen ν_1 und ν_2 dargestellt, sowie die Aggregation mittels $agg_{\prod}$. Die Distanz-ID definiert hierbei einen Identifikator für ein Objektpaar. Betrachtet man die ersten fünf Ränge der aggregierten Distanzen, so sieht man, dass die top-5-Objekte von Distanzfunktion d_1 komplett mit denen der Aggregation übereinstimmen, während bei Distanzfunktion d_2 lediglich zwei Werte in der Rangfolge der aggregierten Distanzen auftreten. Gleiches gilt für die Ränge 6–10. Damit zeigt die Distanzfunktion d_1 eine Dominanz gegenüber der Distanzfunktion d_2. Schaut man sich noch einmal die Intervalle der Verteilung ν_1 und ν_2 an, so zeigt sich, dass die Dominanz dem großen Unterschied der Verteilungsintervalle (0.7 vs. 0.2) obliegt. Eine Dominanz manifestiert sich also vor allem wenn eine große Differenz zwischen den jeweiligen Intervallen der Distanzverteilungen liegt.

3.3 Messung der Dominanz

Um die Überwertung aus unserem Beispiel und somit die Dominanz zu quantifizieren, wird die Korrelation zwischen den Distanzen von d_1 (d_2) und der aggregierten Distanzen aus d_{agg} bestimmt. Zur Berechnung der Korrelation können mehrere Verfahren genutzt werden. Verwendet man wie im obigen Beispiel nur die Ränge, so bietet sich Spearmans Rangkorrelationskoeffizient an [13].

$$\begin{aligned} \rho(A, B) &= \frac{Cov(\text{Rang}(A), \text{Rang}(B))}{\sigma_{\text{Rang}(A)} * \sigma_{\text{Rang}(B)}} \text{ mit} \\ Cov(X, Y) &= E\left[(X - \mu_x) * (Y - \mu_y)\right] \end{aligned} \tag{1}$$

Hierbei sei $Cov(X, Y)$ die über den Erwartungswert von X und Y definierte Kovarianz. Bezogen auf das vorherige Beispiel erhalten wir eine Korrelation nach Spearman für d_1 von $\rho_1 = 0.94$ und für d_2 $\rho_2 = 0.45$. Die Differenz der Korrelationswerte liegt dabei bei $\rho_1 - \rho_2 = 0.49$. Ab $\epsilon = 0.2$ lässt sich eine Überbewertung einer Distanzfunktion feststellen. Somit haben wir mit $\rho_1 - \rho_2 = 0.49 > 0.2$ eine starke Überbewertung von d_1 gegenüber d_2 in Bezug auf das Aggregationsergebnis gezeigt.
Durch die Verwendung der Rangwerte gibt es allerdings einen Informationsverlust. Eine alternative Berechnung ohne Informationsverlust wäre durch Pearsons Korrelationskoeffizienten möglich [10]. Genügen die Ranginformationen, dann bietet Spearmans Rangkorrelationskoeffizient durch eine geringere Anfälligkeit gegenüber Ausreißern an [14].
Bisher haben wir die Korrelation zwischen den aggregierten Werten und denen aus je einer Distanzverteilung verglichen. Um direkt eine Beziehung zwischen zwei verschiedenen Distanzverteilungen bzgl. einer aggregierten Verteilung zu bestimmen, werden zunächst die zwei Korrelationswerte ρ_1 und ρ_2 der Distanzfunktionen d_1 und d_2 bzgl. ihres Einflusses auf das Aggregationsergebnis graphisch dargestellt [6]. Hierzu werden die jeweiligen Werte der Korrelation als Punkte in $[-1, 1]^2$ definiert. Für eine gleichmäßige Beeinflussung des Aggregationsergebnisses sollten sich die Punkte auf der Diagonalen durch den Koordinatenursprung mit

[2]Das Problem der Dominanz tritt auch bei anderen Aggregationsfunktionen wie Summe, Mittelwert etc. auf und kann zusätzlich eine Dominanz hervorrufen, z.B. bei der Minimum/Maximumfunktion.

Rang	d_1	Distanz-ID	d_2	Distanz-ID	$agg_{\prod}$	Distanz-ID
1	0.729	1	0.487	8	0.347	8
2	0.712	8	0.481	5	0.285	4
3	0.694	4	0.426	10	0.266	1
4	0.547	9	0.425	7	0.235	5
5	0.488	5	0.421	3	0.205	9
6	0.473	7	0.411	4	0.201	7
7	0.394	10	0.375	9	0.168	10
8	0.351	3	0.367	6	0.148	3
9	0.337	2	0.365	1	0.112	6
10	0.306	6	0.316	2	0.106	2

Figure 3: Dominanzproblem bei unterschiedlichen Verteilungen

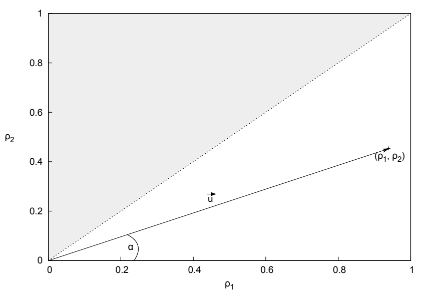


Figure 4: Graphische Darstellung der Korrelation ρ_1 und ρ_2 auf das Aggregationsergebnis

dem Anstieg $m = 1$ befinden. Wir bezeichnen diese Gerade als Kalibrierungslinie. Für unser Beispiel genügt es, nur positive Korrelationswerte zu betrachten. Damit kennzeichnen alle Punkte unterhalb dieser Linie einen größeren Einfluss durch d_1. Analog gilt bei allen Punkten oberhalb dieser Linie (grau schraffierter Bereich) eine größere Beeinflussung durch d_2. Abbildung 4 zeigt graphisch die Korrelation für unser Beispiel von ρ_1 und ρ_2 auf das Aggregationsergebnis. Um die Abweichung vom gewünschten Zustand zu bestimmen, ermitteln wir den Winkel zwischen dem Ortsvektor $\vec{u} = (\rho_1, \rho_2)^T$ durch den Punkt (ρ_1, ρ_2) und der horizontalen Koordinatenachse [6]. Der Winkel α ergibt sich dann durch $\alpha = \arctan\left(\frac{\rho_2}{\rho_1}\right)$ Dieser Winkel liegt zwischen $[0, \frac{\Pi}{2}]$, während die Kalibrierungslinie mit der horizontalen Achse einen Winkel von $\frac{\Pi}{4}$ einschließt. Für eine vorzeichenbehaftete Kennzeichnung der Überbewertung sollen nun alle Korrelationspunkte unterhalb der Kalibrierungslinie einen positiven Wert und alle Korrelationspunkte oberhalb einen negativen Wert erhalten. Für ein Maß der Dominanz definieren wir nun folgende Berechnung [6]:

$$Cal_{err}(\delta^i, \delta^j, \delta^{agg}) = 1 - \frac{4}{\pi}\arctan\left(\frac{Corr(\delta^j, \delta^{agg})}{Corr(\delta^i, \delta^{agg})}\right) \quad (2)$$

Hierbei definiert $Corr(X, Y)$ ein geeignetes Korrelationsmaß, in unserem Fall der Rangkorrelationskoeffizient von Spearman. Wir bezeichnen dieses Maß als Kalibrierungsfehler, wobei ein Fehler von 0 bedeutet, dass es keine Dominanz gibt und somit beide Distanzfunktionen gleichermaßen in das Aggregationsergebnis einfließen. Der Wertebereich des Kalibrierungsfehlers Cal_{err} liegt in $[-1, 1]$. Für unser Beispiel erhalten wir unter Verwendung von Spearmans Rangkorrelationskoeffizienten $Cal_{err}(d_1, d_2, d_{agg}) = 0.43$, womit erkennbar ist, dass d_1 das Aggregationsergebnis stärker beeinflusst als d_2.

DEFINITION 3 (*Kalibrierungsfehler*). *Ein Kalibrierungsfehler liegt vor, wenn es eine Dominanz einer Distanzfunktion d_1 gegenüber d_2 gibt, d.h. die Korrelationswerte nicht auf der Kalibrierungslinie liegen. Entsprechend sind zwei Verteilungen von Distanzwerten kalibriert, wenn kein Kalibrierungsfehler auftritt.*

Analog zur Definition eines ϵ-Wertes zeigte eine empirische Untersuchung für einen Wert $\tau \geq 0.1$ eine ungleichmäßige Auswirkung auf das Aggregationsergebnis.

3.4 Zusammenfassung

Wir haben in diesem Kapitel gezeigt wann ein Dominanzproblem auftritt und wie groß der Einfluss auf das Aggregationsergebnis sein kann. Mit der Verwendung von Gleichung (2) ist es nun möglich den Grad des Dominanzproblems bzw. den Kalibrierungsfehler messen zu können. Ein Hauptgrund für das Auftreten des Dominanzproblem liegt in der Verteilung der Distanzen. Sind die Intervalle, in denen die Distanzen liegen unterschiedlich groß, so ist die Dominanz einer Eigenschaft unvermeidbar. Können diese Intervalle der Distanzverteilungen aneinander angeglichen werden ohne dabei die Rangfolge zu verletzen, so könnte dies das Dominanzproblem lösen. Weiterhin ermöglicht das Maß des Kalibrierungsfehlers die Evaluation von Normalisierungsansätzen.

4. STAND DER TECHNIK

Die Aggregation auf Basis mehrerer Eigenschaften ist ein weit verbreitetes Feld. Es gibt bereits eine Vielzahl von Arbeiten die sich mit dem Thema der *Score-Normalization* beschäftigten. Die Evaluierung solcher Ansätze erfolgt in vielen Fällen, vor allem im Bereich des IR, direkt über die Auswertung der Qualität der Suchergebnisse anhand verschiedener Dokumentenkollektionen, z.B. TREC-Kollektionen[3]. Dieses Vorgehen liefert aber kaum Anhaltspunkte, warum sich einige Normalisierungsansätze besser für bestimmte Anwendungen eignen als andere [6].
Betrachten wir zunächst verschiedene lineare Normalisierungen der Form normalize$(\delta) = y_{min} + \frac{\delta - x_{min}}{x_{max} - x_{min}}(y_{max} - y_{min})$ [15], wobei die Bezeichnungen x_{min}, x_{max}, y_{min} und y_{max} verschiedene Normalisierungsparameter darstellen. Tabelle 1 stellt einige solcher linearer Ansätze dar [15, 5, 9, 6].

Name	y_{min}	y_{max}	x_{min}	x_{max}
Min-Max	0	1	$min(\delta)$	$max(\delta)$
Fitting	$0 < a$	$a < b < 1$	$min(\delta)$	$max(\delta)$
ZUMV	0	1	μ_δ	σ_δ
ZUMV2	2	3	μ_δ	σ_δ
MAD	0	1	Median(δ)	$MAD(\delta)$

Table 1: Parameter lin. Normalisierungsverfahren

Neben den linearen Normalisierungsfunktionen gibt es auch zahlreiche nicht-lineare. Darunter fallen z.B. der tanh-Estimator [4] und die Double-Sigmoid [2] Normalisierung.

[3]Text Retrieval Conference (`http://trec.nist.gov/`)

Beide sind den linearen Verfahren jedoch sehr ähnlich. Avampatzis und Kamps stellen in [1] drei verschieden Normalisierungsverfahren vor, die alle auf der Annahme basieren, dass sich ein Score-Wert eine Summe aus einer Signal und einer Noise-Komponente zusammensetzen, wobei das Verhältnis der Summanden nur von dem Gesamt-Score abhängt [6, 1].
Für das Problem der Dominanz lässt sich einfach zeigen, dass diese Ansätze keinen direkten Einfluss auf die Distanzverteilung haben. Es werden maximal zwei statistische Merkmale (Minimum, Maximum, Median etc.) genutzt, um eine Normalisierung durchzuführen [9, 7]. Auch wenn diese Ansätze auf einigen Testkollektionen Verbesserungen in der Retrieval-Qualität erreichten, so kann nicht sichergestellt werden, dass diese Ansätze allgemein zu einer Verbesserung des Dominanzproblems beitragen. Besonders problematisch sind Ausreißer in den Verteilungen, die das Dominanzproblem bei einer Aggregation sogar noch verstärken können. Ebenfalls problematisch sind Aggregationen auf unterschiedlichen Distanzverteilungen, z.B. Normal- und Gleichverteilungen.
Es gibt allerdings auch Ansätze, die die Distanzverteilung als Grundlage zur Normalisierung heranziehen. Hierbei wird versucht die Distanzen aus unterschiedlichen Quellen so abzubilden, dass sie möglichst exakt gleiche Verteilungen besitzen. Die Ansätze von Manmatha [8] und Fernandez [3] analysieren dabei das probabilistische Verhalten von Suchmaschinen unter der Annahme, dass relevante Dokumente eine Normalverteilung und irrelevante eine exponentielle Verteilung besitzen. Diese Ansätze bieten zwar eine optimierte Normierung, erfordern aber gleichzeitig Zusatzinformation (z.B. über die Relevanz von Textdokumenten), die in vielen Anwendungsfällen gar nicht vorhanden sind.

5. ZUSAMMENFASSUNG UND AUSBLICK

In dieser Arbeit wurde ein Verfahren vorgestellt um die Dominanz unterschiedlicher Eigenschaften messen zu können. Hierzu wurde zunächst der Begriff der Dominanz und dessen Auswirkung untersucht. Anschließend wurde auf Basis eines von Distanzverteilungen ein Maß vorgestellt, mit dessen Hilfe der Grad der Dominanz bestimmt werden kann. Dies ermöglicht uns eine Überbewertung zu erkennen und die Qualität eines Normalisierungsverfahrens zu evaluieren. Die in dieser Arbeit vorgestellten Normalisierungsverfahren wiesen jedoch einige Schwächen auf. Hinzu kommt, dass die algebraischen Eigenschaften der zugrunde liegenden Distanzfunktionen gänzlich ungeachtet blieben (Problem fehlender Metrikeigenschaften). In zukünftigen Arbeiten soll daher ein Ansatz entwickelt werden, der beide Probleme gleichermaßen zu lösen versucht. Hierzu soll ein Verfahren der multivariaten Statistik, die multidimensionale Skalierung, verwendet werden. Zusätzlich sollen die Auswirkungen unterschiedlicher Normalisierungsansätze auf Dominanz und (Retrieval-) Qualität untersucht werden.

6. REFERENCES

[1] A. Arampatzis and J. Kamps. A signal-to-noise approach to score normalization. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 797–806, New York, NY, USA, 2009. ACM.

[2] R. Cappelli, D. Maio, and D. Maltoni. Combining fingerprint classifiers. In *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings*, pages 351–361, 2000.

[3] M. Fernández, D. Vallet, and P. Castells. Probabilistic score normalization for rank aggregation. In *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006, London*, volume 3936 of *Lecture Notes in Computer Science*, pages 553–556. Springer, 2006.

[4] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics - The Approach Based on Influence Functions.* Wiley, 1986. missing.

[5] A. Jain, K. Nandakumar, and A. Ross. Score normalization in multimodal biometric systems. *Pattern Recogn.*, 38(12):2270–2285, Dec. 2005.

[6] R. Kuban. Analyse von Kalibrierungsansätzen für die CQQL-Auswertung. Bachelor's thesis, University of Cottbus-Senftenberg, Germany, Oct. 2012.

[7] J. H. Lee. Analyses of multiple evidence combination. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267–276, New York, NY, USA, 1997. ACM Press.

[8] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267–275, New York, NY, USA, 2001. ACM Press.

[9] M. Montague and J. A. Aslam. Relevance score normalization for metasearch. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 427–433, New York, NY, USA, 2001. ACM.

[10] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42:59–66, 1988.

[11] H. Samet. *Foundations of Multidimensional And Metric Data Structures.* Morgan Kaufmann, 2006/08/08/ 2006.

[12] L. A. Shalabi and Z. Shaaban. Normalization as a preprocessing engine for data mining and the approach of preference matrix. In *DepCoS-RELCOMEX*, pages 207–214. IEEE Computer Society, 2006.

[13] C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:88–103, 1904.

[14] K. Varmuza and P. Filzmoser. *Introduction to Multivariate Statistical Analysis in Chemometrics.* Taylor & Francis, 2008.

[15] S. Wu, F. Crestani, and Y. Bi. Evaluating score normalization methods in data fusion. In *Proceedings of the Third Asia Conference on Information Retrieval Technology*, AIRS'06, pages 642–648, Berlin, Heidelberg, 2006. Springer-Verlag.

[16] D. Zellhöfer. Eliciting Inductive User Preferences for Multimedia Information Retrieval. In W.-T. Balke and C. Lofi, editors, *Proceedings of the 22nd Workshop "Grundlagen von Datenbanken 2010"*, volume 581, 2010.

PEL: Position-Enhanced Length Filter for Set Similarity Joins

Willi Mann
Department of Computer Sciences
Jakob-Haringer-Str. 2
Salzburg, Austria
wmann@cosy.sbg.ac.at

Nikolaus Augsten
Department of Computer Sciences
Jakob-Haringer-Str. 2
Salzburg, Austria
nikolaus.augsten@sbg.ac.at

ABSTRACT

Set similarity joins compute all pairs of similar sets from two collections of sets. Set similarity joins are typically implemented in a filter-verify framework: a filter generates candidate pairs, possibly including false positives, which must be verified to produce the final join result. Good filters produce a small number of false positives, while they reduce the time they spend on hopeless candidates. The best known algorithms generate candidates using the so-called prefix filter in conjunction with length- and position-based filters.

In this paper we show that the potential of length and position have only partially been leveraged. We propose a new filter, the *position-enhanced length filter*, which exploits the matching position to incrementally tighten the length filter; our filter identifies hopeless candidates and avoids processing them. The filter is very efficient, requires no change in the data structures of most prefix filter algorithms, and is particularly effective for foreign joins, i.e., joins between two different collections of sets.

1. INTRODUCTION

The *set similarity join* computes all pairs of similar sets from two collections of sets. The similarity is assessed using a set similarity function, e.g., set overlap, Jaccard, or Cosine similarity, together with a threshold. A pair of sets is in the join result if the similarity exceeds the threshold.

Set similarity joins have many interesting applications ranging from near duplicate detection of Web documents to community mining in social networks [9]. The set elements are called *tokens* [3] and are often used to represent complex objects, e.g., strings (q-grams [11]) or trees (pq-grams [2]).

The best algorithms for set similarity joins are based on an inverted list index and the so-called *prefix filter* [5]. The prefix filter operates on sorted sets and rejects candidate pairs that have no overlap in a (short) prefix. Only the prefix must be indexed, which leads to substantial savings in space and runtime. However, for frequent tokens, the prefix filter produces a large number of candidates. Recent developments in the field leverage the position of the matching tokens between two prefixes (positional filter) and the number of remaining tokens in the overall set (length filter) to further reduce the number of candidates.

This paper proposes a new filter, the *position-enhanced length filter* (PEL), which tightens the length filter based on the position of the current token match. In previous work, position and length information have only partially been exploited. PEL fully leverages position and length to achieve additional pruning power. As a key feature, PEL does not require changes in the prefix filter index, but is applied on top of previous algorithms at almost no cost. In our experiments we show that PEL is particularly effective for foreign joins. PEL also performs well for self joins over large collections of small sets.[1]

The remaining paper is organized as follows: Section 2 introduces the set similarity join, provides background material, and an in-depth analysis of filtering techniques based on position and length. Our novel position-enhanced length filter (PEL) is introduced in Section 3. We empirically evaluate our technique and demonstrate its effectiveness on real world data in Section 4. In Section 5 we survey related work and finally draw conclusions in Section 6.

2. BACKGROUND

We revisit candidate generation, candidate reduction, and efficient verification techniques discussed in literature. The main concepts of this section are summarized in Figure 1.

2.1 Candidate Generation

We shorty explain the prefix filter, translate normalized thresholds to overlap thresholds, revisit length- and position-based filter conditions, and discuss prefixes in index implementations of set similarity joins.

Prefix Filter. The fastest set similarity joins are based on the prefix filter principle [5]: A pair of sorted sets s_0, s_1 can only meet an overlap threshold t_O, i.e., $|s_0 \cap s_1| \geq t_O$, if there is a non-zero overlap in the prefixes of the sets. The prefixes are of length $|s_0| - t_O + 1$ for s_0 and $|s_1| - t_O + 1$ for s_1. The set similarity join proceeds in three steps: (a) index the prefixes of one join partner, (b) probe the prefixes of the other join partner against the index to generate candidates, (c) verify the candidates by inspecting the full sets.

Threshold for normalized set overlap. Normalized set similarity measures take the set sizes into account. For

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

[1]In a self join, both input relations are identical, which cannot be assumed in a foreign join.

Table 1: Set similarity functions and related definitions, extending [7, Table 1] by new pmaxsize.

	Similarity function	**minoverlap**(t, s_0, s_1)	**minsize**(t, s_0)	**pmaxsize**(t, s_0, p_0)	**maxsize**(t, s_0)
Jaccard	$J(s_0, s_1) = \frac{\lvert s_0 \cap s_1 \rvert}{\lvert s_0 \cup s_1 \rvert}$	$\frac{t}{1+t}(\lvert s_0\rvert + \lvert s_1\rvert)$	$t\,\lvert s_0\rvert$	$\frac{\lvert s_0\rvert-(1+t)\cdot p_0}{t}$	$\frac{\lvert s_0\rvert}{t}$
Cosine	$C(s_0, s_1) = \frac{\lvert s_0 \cap s_1\rvert}{\sqrt{\lvert s_0\rvert \cdot \lvert s_1\rvert}}$	$t\sqrt{\lvert s_0\rvert \cdot \lvert s_1\rvert}$	$t^2\,\lvert s_0\rvert$	$\frac{(\lvert s_0\rvert-p_0)^2}{\lvert s_0\rvert \cdot t^2}$	$\frac{\lvert s_0\rvert}{t^2}$
Dice	$D(s_0, s_1) = \frac{2\cdot(\lvert s_0 \cap s_1\rvert)}{\lvert s_0\rvert+\lvert s_1\rvert}$	$\frac{t(\lvert s_0\rvert+\lvert s_1\rvert)}{2}$	$\frac{t\,\lvert s_0\rvert}{2-t}$	$\frac{(2-t)\cdot\lvert s_0\rvert-2p_0}{t}$	$\frac{(2-t)\lvert s_0\rvert}{t}$
Overlap	$O(s_0, s_1) = \lvert s_0 \cap s_1\rvert$	t	t	∞	∞

the purpose of set similarity joins, the normalized threshold is translated into an overlap threshold (called **minoverlap**). The overlap threshold may be different for each pair of sets. For example, for the Cosine threshold t_C (join predicate $|s_0 \cap s_1|/\sqrt{|s_0| \cdot |s_1|} \geq t_C$), **minoverlap**$(t_C, s_0, s_1) := t_C \cdot \sqrt{|s_0| \cdot |s_1|}$. Table 1 lists the definitions of well-known set similarity functions with the respective overlap thresholds.

Length filter. For a given threshold t and a reference set s_0, the size of the matching partners must be within the interval [**minsize**(t, s_0), **maxsize**(t, s_0)] (see Table 1). This was first observed for the PartEnum algorithm [1] and was later called the *length filter*. Example: $|s_0| = 10$, $|s_1| = 6$, $|s_2| = 16$, Cosine threshold $t_C = 0.8$. Since **minoverlap**$(t_C, s_0, s_1) = 6.1 > |s_1|$ we conclude (without inspecting any set element) that s_0 cannot reach threshold t_C with s_1. Similarly, **minoverlap**$(t_C, s_0, s_2) = 10.1$, thus s_2 is too large to meet the threshold with s_0. In fact, **minsize**$(t_C, s_0) = 6.4$ and **maxsize**$(t_C, s_0) = 15.6$.

Prefix length. The prefix length is $|s_0| - t_O + 1$ for a given overlap threshold t_O and set s_0. For *normalized* thresholds t the prefix length does not only depend on s_0, but also on the sets we compare to. If we compare to s_1, the minimum prefix size of $|s_0|$ is **minprefix**$(t, s_0, s_1) = |s_0| -$ **minoverlap**$(t, s_0, s_1) + 1$. When we index one of the join partners, we do not know the size of the matching partners upfront and need to cover the worst case; this results in the prefix length **maxprefix**$(t, s_0) = |s_0| -$**minsize**$(t, s_0)+1$ [7], which does not depend on s_1. For typical Jaccard thresholds $t \geq 0.8$, this reduces the number of tokens to be processed during the candidate generation phase by 80 % or more.

For self joins we can further reduce the prefix length [12] w.r.t. **maxprefix**: when the index is built on-the-fly in increasing order of the sets, then the indexed prefix of s_0 will never be compared to any set s_1 with $|s_1| < |s_0|$. This allows us to reduce the prefix length to **midprefix**$(t, s_0) = |s_0| -$ **minoverlap**$(t, s_0, s_0) + 1$.

Positional filter. The minimum prefix length for a pair of sets is often smaller than the worst case length, which we use to build and probe the index. When we probe the index with a token from the prefix of s_0 and find a match in the prefix of set s_1, then the matching token may be outside the optimal prefix. If this is the first matching token between s_0 and s_1, we do not need to consider the pair. In general, a candidate pair s_0, s_1 must be considered only if

$$\textbf{minoverlap}(t, s_0, s_1) \leq o + \min\{|s_0| - p_0, |s_1| - p_1\}, \quad (1)$$

where o is the current overlap (i.e., number of matching tokens so far excluding the current match) and p_0 (p_1) is the position of the current match in the prefix of s_0 (s_1); positions start at 0.

Previous work:

- **minoverlap**(t, s_0, s_1): equivalent overlap threshold for Jaccard, Cosine, or Dice threshold t for a pair of sets s_0, s_1.
- **minsize**(t, s_0), **maxsize**(t, s_0): minimum and maximum size of any set that can satisfy threshold t w.r.t. set s_0.
- **maxprefix**$(t, s_0) = |s_0| -$ **minsize**$(t, s_0) + 1$: length of probing prefix
- **midprefix**$(t, s_0) = |s_0| -$ **minoverlap**$(t, s_0, s_0) + 1$: length of indexing prefix for self joins
- **minprefix**$(t, s_0, s_1) = |s_0| -$ **minoverlap**$(t, s_0, s_1) + 1$: length of optimal prefix for a particular pair of sets

Position-enhanced length filter (PEL):

- **pmaxsize**(t, s_0, p_0): new tighter limit for maximum set size based on the probing set position

Figure 1: Overview of functions.

The positional filter is stricter than the prefix filter and is applied on top of it. The pruning power of the positional filter is larger for prefix matches further to right (i.e., when p_0, p_1 increase). Since the prefix filter may produce the same candidate pair multiple times (for each match in the prefix), an interesting situation arises: a pair that passes the positional filter for the first match may not pass the filter for later matches. Thus, the positional filter is applied to pairs that are already in the candidate set whenever a new match is found. To correctly apply the positional filter we need to maintain the overlap value for each pair in the candidate set. We illustrate the positional filter with examples.

Example 1. Set s_0 in Figure 2 is the probing set (prefix length **maxprefix** = 4), s_1 is the indexed set (prefix length **midprefix** = 2, assuming self join). Set s_1 is returned from the index due to the match on g (first match between s_0 and s_1). The required overlap is $\lceil \textbf{minoverlap}_C(0.8, s_0, s_1) \rceil = 8$. Since there are only 6 tokens left in s_1 after the match, the maximum overlap we can get is 7, and the pair is pruned. This is also confirmed by the positional filter condition (1) ($o = 0$, $p_0 = 3$, $p_1 = 1$).

Example 2. Assume a situation similar to Figure 2, but the match on g is the *second* match (i.e., $o = 1$, $p_0 = 3$, $p_1 = 1$). Condition (1) holds and the pair can not be pruned, i.e., it remains in the candidate set.

Example 3. Consider Figure 3 with probing set s_0 and indexed set s_1. The match on token a adds pair (s_0, s_1) to the candidate set. Condition (1) holds for the match on **a** ($o = 0$, $p_0 = 0$, $p_1 = 0$), and the pair is not pruned by the positional filter. For the next match (on e), however, condition (1) does not hold ($o = 1$, $p_0 = 1$, $p_1 = 4$) and the positional filter removes the pair from the candidate set. Thus, the positional filter does not only avoid pairs to enter

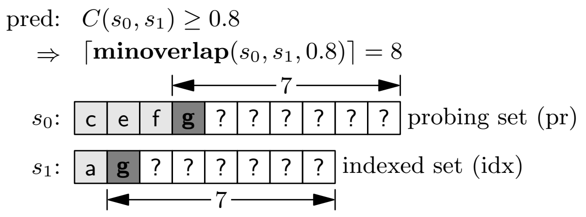

pred: $C(s_0, s_1) \geq 0.8$
$\Rightarrow \lceil$ **minoverlap**$(s_0, s_1, 0.8)\rceil = 8$
7
s_0: c e f g ? ? ? ? ? ? probing set (pr)
s_1: a g ? ? ? ? ? ? indexed set (idx)
7

Figure 2: Sets with matching token In prefix: match impossible due to positions of matching tokens and remaining tokens.

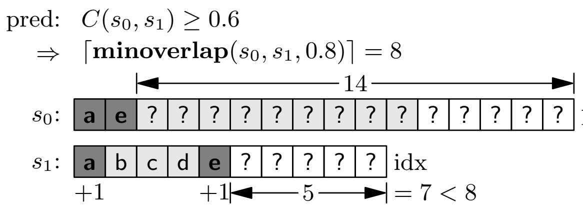

pred: $C(s_0, s_1) \geq 0.6$
$\Rightarrow \lceil$ **minoverlap**$(s_0, s_1, 0.8)\rceil = 8$
14
s_0: a e ? ? ? ? ? ? ? ? ? ? ? ? ? ? pr
s_1: a b c d e ? ? ? ? ? idx
+1
+1
5
= 7 < 8

Figure 3: Sets with two matching tokens: pruning of candidate pair by second match.

the candidate set, but may remove them later.

2.2 Improving the Prefix Filter

The prefix filter often produces candidates that will be removed immediately in the next filter stage, the positional filter (see Example 1). Ideally, such candidates are not produced at all. This issue is addressed in the mpjoin algorithm [7] as outlined below.

Consider condition (1) for the positional filter. We split the condition into two new conditions by expanding the minimum such that the conjunction of the new conditions is equivalent to the positional filter condition:

$$\textbf{minoverlap}(t, s_0, s_1) \leq o + |s_0| - p_0 \tag{2}$$

$$\textbf{minoverlap}(t, s_0, s_1) \leq o + |s_1| - p_1 \tag{3}$$

The mpjoin algorithm leverages condition (2) as follows. The probing sets s_0 are processed in increasing size order, so $|s_0|$ grows monotonically during the execution of the algorithm. Hence, for a specific set s_1, **minoverlap** grows monotonically. We assume $o = 0$ (and justify this assumption later). For a given index entry (s_1, p_1), the right side of condition (2) is constant, while the left side can only grow. After the condition fails to hold for the first time, it will never hold again, and the index list entry is removed. For a given index set s_1, this improvement changes the effective length of the prefix (i.e., the part of the sets where we may detect matches) w.r.t. a probing set s_0 to **minprefix**$(t, s_0, s_1) = |s_1| -$ **minoverlap**$(t, s_0, s_1) + 1$, which is optimal. On the downside, a shorter prefix may require more work in the verification phase: in some cases, the verification can start after the prefix as will be discussed in Section 2.3.

2.3 Verification

Efficient verification techniques are crucial for fast set similarity joins. We revisit a baseline algorithm and two improvements, which affect the verification speed of both false and true positives. Unless explicitly mentioned, the term *prefix* subsequently refers to **maxprefix** (probing set) resp. **midprefix** (indexing set) as discussed in earlier sections.

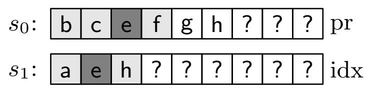


Figure 4: Verification: where to start?

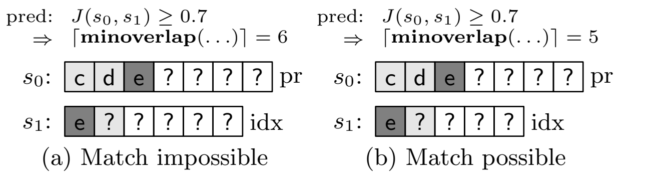

pred: $J(s_0, s_1) \geq 0.7$
$\Rightarrow \lceil$ **minoverlap**$(\ldots)\rceil = 6$
s_0: c d e ? ? ? ? pr
s_1: e ? ? ? ? ? idx
(a) Match impossible
pred: $J(s_0, s_1) \geq 0.7$
$\Rightarrow \lceil$ **minoverlap**$(\ldots)\rceil = 5$
s_0: c d e ? ? ? ? pr
s_1: e ? ? ? ? idx
(b) Match possible

Figure 5: Impossible and possible set sizes based on position in s_0 and the size-dependent minoverlap.

Since the sets are sorted, we compute the overlap in a merge fashion. At each merge step, we verify if the current overlap and the remaining set size are sufficient to achieve the threshold, i.e., we check positional filter condition (1).

(A) Prefix overlap [12]: At verification time we already know the overlap between the two prefixes of a candidate pair. This piece of information should be leveraged. Note that we cannot simply continue verification after the two prefixes. This is illustrated in Figure 4: there is 1 match in the prefixes of s_0 and s_1; when we start verification after the prefixes, we miss token h. Token h occurs after the prefix of s_0 but inside the prefix of s_1. Instead, we compare the last element of the prefixes: for the set with the smaller element (s_0), we start verification after the prefix (g). For the other set (s_1) we leverage the number of matches in the prefix (overlap o). Since the leftmost positions where these matches can appear are the first o elements, we skip o tokens and start at position o (token e in s_1). There is no risk of double-counting tokens w.r.t. overlap o since we start after the end of the prefix in s_0.

(B) Position of last match [7]: A further improvement is to store the position of the last match. Then we start the verification in set s_1 after this position (h in s_1, Figure 4).

Small candidate set vs. fast verification. The positional filter is applied on each candidate pair returned by the prefix filter. The same candidate pair may be returned multiple times for different matches in the prefix. The positional filter potentially removes existing candidate pairs when they appear again (cf. Section 2.1). This reduces the size of the candidate set, but comes at the cost of (a) lookups in the candidate set, (b) deletions from the candidate set, and (c) book keeping of the overlaps for each candidate pair. Overall, it might be more efficient to batch-verify a larger candidate set than to incrementally maintain the candidates; Ribeiro and Härder [7] empirically analyze this trade-off.

3. POSITION-ENHANCED LENGTH FILTERING

In this section, we motivate the *position-enhanced length filter* (PEL), derive the new filter function **pmaxsize**, discuss the effect of PEL on self vs. foreign joins, and show how to apply PEL to previous algorithms.

Motivation. The introduction of the position-enhanced length filter is inspired by examples for positional filtering

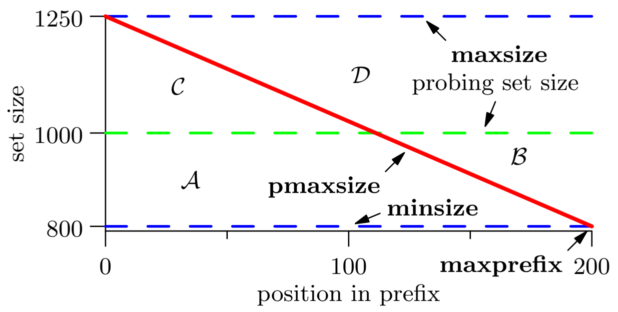


Figure 6: Illustrating possible set sizes.

like Figure 5(a). In set s_1, the only match in the prefix occurs at the leftmost position. Despite this being the leftmost match in s_1, the positional filter removes s_1: the overlap threshold cannot be reached due the position of the match in s_0. Apparently, the position of the token in the probing set can render a match of the index sets impossible, independently of the matching position in the index set. Let us analyze how we need to modify the example such that it passes the positional filter: the solution is to shorten index set s_1, as shown in Figure 5(b). This suggests that some tighter limit on the set size can be derived from the position of the matching token.

Deriving the PEL filter. For the example in Figure 5(a) the first part of the positional filter, i.e., condition (2), does not hold. We solve the equation $\mathbf{minoverlap}(t, s_0, s_1) \leq |s_0| - p_0$ to $|s_1|$ by replacing **minoverlap** with its definition for the different similarity functions. The result is $\mathbf{pmaxsize}(t, s_0, p_0)$, an upper bound on the size of eligible sets in the index. This bound is at the core of the PEL filter, and definitions of **pmaxsize** for various similarity measures are listed in Table 1.

Application of PEL. We integrate the **pmaxsize** upper bound into the prefix filter. The basic prefix filter algorithm processes a probing set as follows: loop over the tokens of the probing set from position $p_0 = 0$ to $\mathbf{maxprefix}(t, s_0) - 1$ and probe each token against the index. The index returns a list of sets (their IDs) which contain this token. The sets in these lists are ordered by increasing size, so we stop processing a list when we hit a set that is larger than $\mathbf{pmaxsize}(t, s_0, p_0)$.

Intuitively, we move half of the positional filter to the prefix filter, where we can evaluate it at lower cost: (a) the value of **pmaxsize** needs to be computed only once for each probing token; (b) we check **pmaxsize** against the size of each index list entry, which is a simple integer comparison. Overall, this is much cheaper than the candidate lookup that the positional filter must do for each index match.

Self Joins vs. Foreign Joins. The PEL filter is more powerful on foreign joins than on self joins. In self joins, the size of the probing set is an upper bound for the set size in the index. For all the similarity functions in Table 1, **pmaxsize** is below the probing set size in less than 50% of the prefix positions. Figure 6 gives an example: The probing set size is 1000, the Jaccard threshold is 0.8, so $\mathbf{minsize}(0.8, 1000) = 800$, $\mathbf{maxsize}(0.8, 1000) = 1250$, and the prefix size is 201. The x-axis represents the position in the prefix, the y-axis represents bounds for the set size of the other set. The region between **minsize** and **maxsize** is the base region. The base region is partitioned into four regions ($\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$) by the probing set size and **pmaxsize**. For foreign joins, our filter reduces the base region to $\mathcal{A}+\mathcal{C}$. If we assume that all set sizes occur equally likely in the individual inverted lists of the index, our filter cuts the number of index list entries that must be processed by 50%. Since the tokens are typically ordered by their frequency, the list length will increase with increasing matching position. Thus the gain of PEL in practical settings can be expected to be even higher. This analysis holds for all parameters of Jaccard and Dice. For Cosine, the situation is more tricky since **pmaxsize** is quadratic and describes a parabola. Again, this is in our favor since the parabola is open to the top, and the curve that splits the base region is below the diagonal.

For self joins, the only relevant regions are $\mathcal{A}$ and $\mathcal{B}$ since the size of the sets is bounded by the probing set size. Our filter reduces the relevant region from $\mathcal{A} + \mathcal{B}$ to $\mathcal{A}$. As Figure 6 illustrates, this reduction is smaller than the reduction for foreign joins. For the similarity functions in Table 1, $\mathcal{B}$ is always less than a quarter of the full region $\mathcal{A} + \mathcal{B}$. In the example, region $\mathcal{B}$ covers about 0.22 of $\mathcal{A} + \mathcal{B}$.

Algorithm 1: AllPairs-PEL(S_p, I, t)

```
Version using pmaxsize for foreign join;
input  : $S_p$ collection of outer sets, $I$ inverted list index
         covering maxprefix of inner sets, $t$ similarity
         threshold
output: $res$ set of result pairs (similarity at least $t$)
foreach $s_0$ in $S_p$ do
    $M = \{\}$; /* Hashmap: candidate set → count */
    for $p_0 \leftarrow 0$ to maxprefix$(t, s_0) - 1$ do
        for $s_1$ in $I_{s_0[p]}$ do
            if $|s_1| <$ minsize$(t, s_0)$ then
                remove index entry with $s_1$ from $I_{s_0[p]}$;
            else if $|s_1| >$ pmaxsize$(t, s_0, p_0)$ then
                break;
            else
                if $M[s_1] = \emptyset$ then
                    $M = M \cup (s_1, 0)$;
                $M[s_1] = M[s_1] + 1$;
        end
    end
    /* Verify() verifies the candidates in $M$ */
    $res = res \cup Verify(s_0, M, t)$;
end
```

Algorithm. Algorithm 1 shows AllPairs-PEL[2], a version of AllPairs enhanced with our PEL filter. AllPairs-PEL is designed for foreign joins, i.e., the index is constructed in a preprocessing step before the join is executed. The only difference w.r.t. AllPairs is that AllPairs-PEL uses $\mathbf{pmaxsize}(t, s_0, p_0)$ instead of $\mathbf{maxsize}(t, s_0)$ in the condition on line 7. The extensions of the algorithms ppjoin and mpjoin with PEL are similar.

An enhancement that is limited to ppjoin and mpjoin is to simplify the positional filter: PEL ensures that no candidate set can fail on the first condition (Equation 2) of the split positional filter. Therefore, we remove the first part of the

[2]We use the *-PEL* suffix for algorithm variants that make use of our PEL filter.

Table 2: Input set characteristics.

	#sets in collection	set size min	set size max	set size avg	# of diff. tokens
DBLP	$3.9 \cdot 10^6$	2	283	12	$1.34 \cdot 10^6$
TREC	$3.5 \cdot 10^5$	2	628	134	$3.4 \cdot 10^5$
ENRON	$5 \cdot 10^5$	1	192 000	298	$7.3 \cdot 10^6$

minimum in the original positional filter (Equation 1), such that the minimum is no longer needed.

Note that the removal of index entries on line 6 is the easiest solution to apply **minsize**, but in real-world scenarios, it only makes sense for a single join to be executed. For a similarity search scenario, we recommend to apply binary search on the lists. For multiple joins with the same indexed sets in a row, we suggest to use an overlay over the index that stores the pointer for each list where to start.

4. EXPERIMENTS

We compare the algorithms AllPairs [4] and mpjoin [7] with and without our PEL extension on both self and foreign joins. Our implementation works on integers, which we order by the frequency of appearance in the collection. The time to generate integers from tokens is not measured in our experiments since it is the same for all algorithms. We also do not consider the indexing time for foreign joins, which is considered a preprocessing step. The use of PEL has no impact on the index construction. The prefix sizes are **maxprefix** for foreign joins and **midprefix** for self joins. For self joins, we include the indexing time in the overall runtime since the index is built incrementally on-the-fly. We report results for Jaccard and Cosine similarity, the results for Dice show similar behavior. Our experiments are executed on the following real-world data sets:

- DBLP[3]: Snapshot (February 2014) of the DBLP bibliographic database. We concatenate authors and title of each entry and generate tokens by splitting on whitespace.
- TREC[4]: References from the MEDLINE database, years 1987–1991. We concatenate author, title, and abstract, remove punctuation, and split on whitespace.
- ENRON[5]: Real e-mail messages published by FERC after the ENRON bankruptcy. We concatenate subject and body fields, remove punctuation, and split on whitespace.

Table 2 lists basic characteristics of the input sets. We conduct our experiments on an Intel Xeon 2.60GHz machine with 128 GB RAM running Debian 7.6 'wheezy'. We compile our code with gcc -O3. Claims about results on "all" thresholds for a particular data set refer to the thresholds $\{0.5, 0.6, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$. We stop tests whose runtime exceeds one hour.

Foreign Joins. For foreign joins, we join a collection of sets with a copy of itself, but do not leverage the fact that the collections are identical. Figures 7(a) and 7(b) show the performance on DBLP with Jaccard similarity threshold 0.75 and Cosine similarity 0.85. These thresholds produce result sets of similar size. We observe a speedup of factor 3.5 for AllPairs-PEL over AllPairs with Jaccard, and a speedup of 3.8 with Cosine. For mpjoin to mpjoin-PEL we observe a speedup of 4.0 with Jaccard and 4.2 with Cosine. Thus, the PEL filter provides a substantial speed advantage on these data points. For other Jaccard thresholds and mpjoin vs. mpjoin-PEL, the maximum speedup is 4.1 and the minimum speedup is 1.02. For threshold 0.5, only mpjoin-PEL finishes within the time limit of one hour. Among all Cosine thresholds and mpjoin vs. mpjoin-PEL, the maximum speedup is 4.2 ($t_C = 0.85$), the minimum speedup is 1.14 ($t_C = 0.95$). We only consider Cosine thresholds $t_C \geq 0.75$, because the non-PEL variants exceed the time limit for smaller thresholds. There is no data point where PEL slows down an algorithm. It is also worth noting that AllPairs-PEL beats mpjoin by a factor of 2.7 with Jaccard threshold $t_J = 0.75$ and 3.3 on Cosine threshold $t_C = 0.85$; we observe such speedups also on other thresholds.

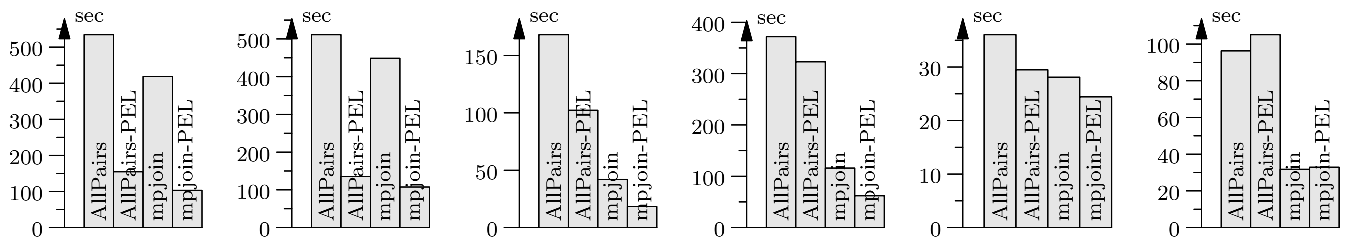
Figure 7(c) shows the performance on TREC with Jaccard threshold $t_J = 0.75$. The speedup for AllPairs-PEL compared to AllPairs is 1.64, and for mpjoin-PEL compared to mpjoin 2.3. The minimum speedup of mpjoin over all thresholds is 1.26 ($t_J = 0.95$), the maximum speedup is 2.3 ($t_J = 0.75$). Performance gains on ENRON are slightly smaller – we observe speedups of 1.15 (AllPairs-PEL over AllPairs), and 1.85 (mpjoin-PEL over mpjoin) on Jaccard threshold $t_J = 0.75$ as illustrated in Figure 7(d). The minimum speedup of mpjoin over mpjoin-PEL is 1.24 ($t_J = 0.9$ and 0.95), the maximum speedup is 2.0 ($t_J = 0.6$).

Figure 8(a) shows the number of processed index entries (i.e., the overall length of the inverted lists that must be scanned) for Jaccard threshold $t_J = 0.75$ on TREC. The number of index entries increases by a factor of 1.67 for AllPairs w.r.t. AllPairs-PEL, and a factor of 4.0 for mpjoin w.r.t. mpjoin-PEL.

Figure 8(b) shows the number of candidates that must be verified for Jaccard threshold $t_J = 0.75$ on TREC. On AllPairs, PEL decreases the number of candidates. This is because AllPairs does not apply any further filters before verification. On mpjoin, the number of candidates increases by 20%. This is due to the smaller number of matches from the prefix index in the case of PEL: later matches can remove pairs from the candidate set (using the positional filter) and thus decrease its size. However, the larger candidate set for PEL does not seriously impact the overall performance: the positional filter is also applied in the verification phase, where the extra candidate pairs are pruned immediately.

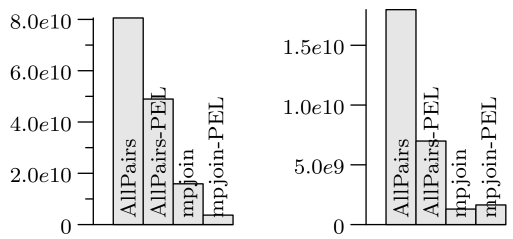
Self joins. Due to space constraints, we only show results for DBLP and ENRON, i.e., the input sets with the smallest and the largest average set sizes, respectively. Figure 7(e) and 7(f) show the performance of the algorithms on DBLP and ENRON with Jaccard threshold $t_J = 0.75$. Our PEL filter provides a speed up of about 1.22 for AllPairs, and 1.17 for mpjoin on DBLP. The maximum speedup we observe is 1.70 (AllPairs-PEL vs. AllPairs, $t_J = 0.6$); for $t_J = 0.95$ there is no speed difference between mpjoin and mpjoin-PEL. On the large sets of ENRON, the performance is worse for AllPairs-PEL because verification takes more time than PEL can save in the probing phase (by reducing the number of processed index entries). There is almost no

[3]`http://www.informatik.uni-trier.de/~Ley/db/`
[4]`http://trec.nist.gov/data/t9_filtering.html`
[5]`https://www.cs.cmu.edu/~enron/`



(a) Foreign join, DBLP, $t_J = 0.75$. (b) Foreign join, DBLP, $t_C = 0.85$. (c) Foreign join, TREC, $t_J = 0.75$ (d) Foreign j., ENRON, $t_J = 0.75$ (e) Self join, DBLP, $t_J = 0.75$ (f) Self join, ENRON, $t_J = 0.75$

Figure 7: Join times.



(a) Number of processed index entries. (b) Number of candidates to be verify.

Figure 8: TREC (foreign join): $t_J = 0.75$

difference between mpjoin and mpjoin-PEL. The maximum increase in speed is 9% (threshold 0.8, mpjoin), the maximum slowdown is 30% (threshold 0.6, AllPairs).

Summarizing, PEL substantially improves the runtime in foreign join scenarios. For self joins, PEL is less effective and, in some cases, may even slightly increase the runtime.

5. RELATED WORK

Sarawagi and Kirpal [8] first discuss efficient algorithms for exact set similarity joins. Chaudhuri et al. [5] propose SSJoin as an in-database operator for set similarity joins and introduce the prefix filter. AllPairs [4] uses the prefix filter with an inverted list index. The ppjoin algorithm [12] extends AllPairs by the positional filter and introduces the suffix filter, which reduces the candidate set before the final verification. The mpjoin algorithm [7] improves over ppjoin by reducing the number of entries returned from the index. AdaptJoin [10] takes the opposite approach and drastically reduces the number of candidates at the expense of longer prefixes. Gionis et al. [6] propose an approximate algorithm based on LSH for set similarity joins. Recently, an SQL operator for the token generation problem was introduced [3].

6. CONCLUSIONS

We presented PEL, a new filter based on the **pmaxsize** upper bound derived in this paper. PEL can be easily plugged into algorithms that store prefixes in an inverted list index (e.g., AllPairs, ppjoin, or mpjoin). For these algorithms, PEL will effectively reduce the number of list entries that must be processed. This reduces the overall lookup time in the inverted list index at the cost of a potentially larger candidate set. We analyzed this trade-off for foreign joins and self joins. Our empirical evaluation demonstrated that the PEL filter improves performance in almost any foreign join and also in some self join scenarios, despite the fact that it may increase the number of candidates to be verified.

7. REFERENCES

[1] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *Proc. VLDB*, pages 918 – 929, 2006.

[2] N. Augsten, M. H. Böhlen, and J. Gamper. The *pq*-gram distance between ordered labeled trees. *ACM TODS*, 35(1), 2010.

[3] N. Augsten, A. Miraglia, T. Neumann, and A. Kemper. On-the-fly token similarity joins in relational databases. In *Proc. SIGMOD*, pages 1495 – 1506. ACM, 2014.

[4] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. *WWW*, 7:131 – 140, 2007.

[5] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *Proc. ICDE*, page 5. IEEE, 2006.

[6] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. VLDB*, pages 518–529, 1999.

[7] L. A. Ribeiro and T. Härder. Generalizing prefix filtering to improve set similarity joins. *Information Systems*, 36(1):62 – 78, 2011.

[8] S. Sarawagi and A. Kirpal. Efficient set joins on similarity predicates. In *Proc. SIGMOD*, pages 743 – 754. ACM, 2004.

[9] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: A large-scale study in the orkut social network. In *Proc. SIGKDD*, pages 678 – 684. ACM, 2005.

[10] J. Wang, G. Li, and J. Feng. Can we beat the prefix filtering?: An adaptive framework for similarity join and search. In *Proc. SIGMOD*, pages 85 – 96. ACM, 2012.

[11] C. Xiao, W. Wang, and X. Lin. Ed-Join: An efficient algorithm for similarity joins with edit distance constraints. In *Proc. VLDB*, 2008.

[12] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *ACM TODS*, 36(3):15, 2011.