SIGMOD Officers, Committees, and Awardees

Chair	Vice-Chair	Secretary/Treasurer
Donald Kossmann Systems Group ETH Zürich Cab F 73 8092 Zuerich SWITZERLAND +41 44 632 29 40 <donaldk AT inf.ethz.ch>	Anastasia Ailamaki School of Computer and Communication Sciences, EPFL EPFL/IC/IIF/DIAS Station 14, CH-1015 Lausanne SWITZERLAND +41 21 693 75 64 <natassa AT epfl.ch>	Magdalena Balazinska Computer Science & Engineering University of Washington Box 352350 Seattle, WA USA +1 206-616-1069 <magda AT cs.washington.edu>

SIGMOD Executive Committee:
Donald Kossmann (Chair), Anastasia Ailamaki (Vice-Chair), Magdalena Balazinska, K. Selçuk Candan, Yanlei Diao, Curtis Dyreson, Christian Jensen, Yannis Ioannidis, and Tova Milo.

Advisory Board:
Raghu Ramakrishnan (Chair, Microsoft), Amr El Abbadi, Serge Abiteboul, Ricardo Baeza-Yates, Phil Bernstein, Elisa Bertino, Mike Carey, Surajit Chaudhuri, Christos Faloutsos, Alon Halevy, Joe Hellerstein, Renée Miller, C. Mohan, Beng-Chin Ooi, Z. Meral Ozsoyoglu, Sunita Sarawagi, Min Wang, and Gerhard Weikum.

SIGMOD Information Director:
Curtis Dyreson, Utah State University <curtis.dyreson AT usu.edu>

Associate Information Directors:
Manfred Jeusfeld, Georgia Koutrika, Wim Martens, Mirella Moro, Rachel Pottinger, and Jun Yang.

SIGMOD Record Editor-in-Chief:
Yanlei Diao, University of Massachusetts Amherst <yanlei AT cs.umass.edu>

SIGMOD Record Associate Editors:
Pablo Barceló, Vanessa Braganholo, Marco Brambilla, Chee Yong Chan, Rada Chirkova, Anish Das Sarma, Alkis Simitsis, Nesime Tatbul, and Marianne Winslett.

SIGMOD Conference Coordinator:
K. Selçuk Candan, Arizona State University

PODS Executive Committee: Rick Hull (Chair, IBM Research), Michael Benedikt, Wenfei Fan, Martin Grohe, Maurizio Lenzerini, Jan Paradaens.

Sister Society Liaisons:
Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment), Christian Jensen (IEEE TKDE).

Awards Committee:
Umesh Dayal (Chair, Hitachi America Ltd.), Elisa Bertino, Surajit Chaudhuri, Masaru Kitsuregawa, and Maurizio Lenzerini.

Jim Gray Doctoral Dissertation Award Committee:
Tova Milo (Co-Chair, Tel Aviv University), Timos Sellis (Co-Chair, RMIT University), Ashraf Aboulnaga, Sudipto Das, Juliana Freire, Minos Garofalakis, Dan Suciu, Kian-Lee Tan.

[Last updated : December 30th, 2014]

SIGMOD Officers, Committees, and Awardees (continued)

SIGMOD Edgar F. Codd Innovations Award

For innovative and highly significant contributions of enduring value to the development, understanding, or use of database systems and databases. Formerly known as the "SIGMOD Innovations Award", it now honors Dr. E. F. (Ted) Codd (1923 - 2003) who invented the relational data model and was responsible for the significant development of the database field as a scientific discipline. Recipients of the award are the following:

Michael Stonebraker (1992)
Jim Gray (1993)
Philip Bernstein (1994)
David DeWitt (1995)
C. Mohan (1996)
David Maier (1997)
Serge Abiteboul (1998)
Hector Garcia-Molina (1999)
Rakesh Agrawal (2000)
Rudolf Bayer (2001)
Patricia Selinger (2002)
Don Chamberlin (2003)
Ronald Fagin (2004)
Michael Carey (2005)
Jeffrey D. Ullman (2006)
Jennifer Widom (2007)
Moshe Y. Vardi (2008)
Masaru Kitsuregawa (2009)
Umeshwar Dayal (2010)
Surajit Chaudhuri (2011)
Bruce Lindsay (2012)
Stefano Ceri (2013)
Martin Kersten (2014)

SIGMOD Contributions Award

For significant contributions to the field of database systems through research funding, education, and professional services. Recipients of the award are the following:

Maria Zemankova (1992)
Gio Wiederhold (1995)
Yahiko Kambayashi (1995)
Jeffrey Ullman (1996)
Avi Silberschatz (1997)
Won Kim (1998)
Raghu Ramakrishnan (1999)
Michael Carey (2000)
Laura Haas (2000)
Daniel Rosenkrantz (2001)
Richard Snodgrass (2002)
Michael Ley (2003)
Surajit Chaudhuri (2004)
Hongjun Lu (2005)
Tamer Özsu (2006)
Hans-Jörg Schek (2007)
Klaus R. Dittrich (2008)
Beng Chin Ooi (2009)
David Lomet (2010)
Gerhard Weikum (2011)
Marianne Winslett (2012)
H.V. Jagadish (2013)
Kyu-Young Whang (2014)

SIGMOD Jim Gray Doctoral Dissertation Award

SIGMOD has established the annual SIGMOD Jim Gray Doctoral Dissertation Award to *recognize excellent research by doctoral candidates in the database field.* Recipients of the award are the following:

- **2006** *Winner*: Gerome Miklau, University of Washington. *Runners-up*: Marcelo Arenas and Yanlei Diao.
- **2007** *Winner*: Boon Thau Loo, University of California at Berkeley. *Honorable Mentions*: Xifeng Yan and Martin Theobald.
- **2008** *Winner*: Ariel Fuxman, University of Toronto. *Honorable Mentions*: Cong Yu and Nilesh Dalvi.
- **2009** *Winner*: Daniel Abadi, MIT. *Honorable Mentions*: Bee-Chung Chen and Ashwin Machanavajjhala.
- **2010** *Winner:* Christopher Ré, University of Washington. *Honorable Mentions*: Soumyadeb Mitra and Fabian Suchanek.
- **2011** *Winner*: Stratos Idreos, Centrum Wiskunde & Informatica. *Honorable Mentions*: Todd Green and Karl Schnaitterz.
- **2012** *Winner*: Ryan Johnson, Carnegie Mellon University. *Honorable Mention*: Bogdan Alexe.
- **2013** *Winner*: Sudipto Das, University of California, Santa Barbara. *Honorable Mention*: Herodotos Herodotou and Wenchao Zhou.
- **2014** *Winners*: Aditya Parameswaran, Stanford University, and Andy Pavlo, Brown University.

A complete listing of all SIGMOD Awards is available at: **http://www.sigmod.org/awards/**

[Last updated : December 30th, 2014]

Editor's Notes

Welcome to the December 2014 issue of the ACM SIGMOD Record!

The issue opens with a Database Principles article by Amsterdamer and Milo on "Foundations of Crowd Data Sourcing." Crowdsourcing has been a popular approach to harnessing data and processing data in domains where human contributions to these tasks are highly valuable, such as websites of reviews and ratings. This article provides a timely survey of important recent work that offers sound scientific foundations for crowd-sourcing. More specifically, it addresses the challenges in modeling crowd sourcing for two main tasks: (1) harvesting data, for which a crowd mining approach is proposed to identify statistically significant patterns within the habits and preferences of the crowd, and analyzed with complexity results; (2) processing data with the help of the crowd, for which classic database operators, including top-k and group-by, are analyzed with theoretical results in this unique problem setting.

The Research and Vision Articles Column features two articles. The first one, by Zliobaite et al, presents a vision of hardware-driven design of low-energy data analysis algorithms. It begins by surveying the recent trend in hardware development, pointing to the need of designing algorithms that minimize communication (memory accesses and data transfer between cores) and related energy consumption. It then considers a range of data mining algorithms as example data analysis tasks, and proposes to make these algorithms more energy-efficient by using more efficient implementations of primitive operations. This is an interesting position paper on integrated algorithm design and hardware design. The second article, by Lissandrini et al., argues that exploitation of information graphs can lead to novel query answering capabilities that go beyond the existing capabilities of keyword search. In particular, it focuses on *exemplar queries* on information graphs, a new querying paradigm where a user keyword query on an information graph is treated as an example from a desired result set and the system infers other elements of the result set via similarity search and ranking.

We are pleased to resume the Distinguished Profiles column with two featured articles in this issue. The first article features the interview with Kian-Lee Tan, who is a Provost's Chair Professor of Computer Science at the National University of Singapore (NUS) and the Vice Dean for Research in the School of Computing. In this interview, Kian-Lee shares with us his past and recent research projects, his view on how to build a strong database group at NUS (without pushing students too hard), and time management tips that allowed him to balance work, family, and church activities. The second interview features Sudipto Das, who graduated from the University of California Santa Barbara and won the 2013 SIGMOD Jim Gray Doctoral Dissertation Award. Sudipto talks about his dissertation research on "Scalable and Elastic Transactional Data Stores for Cloud Computing Platforms," as well as how to deal with ups and downs in PhD studies.

In the Research Centers Column, Bentayeb et al. outline the research agenda at the ERIC laboratory in Lyon, France, which includes two research teams: decision support information systems (SID) and data mining and decision (DMD). The SID team addresses a range of applications and research issues in data warehousing and OLAP, including the integration of complex data sources into an Active XML repository, exploiting active rules and mining logged events to automate integration tasks, supporting OLAP on textual data, and developing a cloud computing environment for big data warehousing and OLAP. The DMD team aims to create new systems, models, and algorithms for data mining and decision making, including efforts on topic modeling for dealing with textual datasets extracted from the Web and leveraging topological graphs to design metrics well-suited for machine learning.

In the Industry Perspectives Column, Ellis and his colleagues at IBM Research present the viewpoint that much work has been done to process big data, but more needs to be done to *understand* big data. As an

effort to bridge the gap, they present Helix, a system for guided exploration of big data. Helix provides a unified view of sources, ranging from spreadsheets and XML files with no schema, to RDF graphs and relational data with well-defined schemas. Helix users explore these heterogeneous data sources through a combination of keyword searches and navigation of linked web pages. This article also presents a set of real-world usage scenarios and the lessons learned in the course of developing Helix.

This issue features an event report, by Pedersen, Castellanos, and Dayal, on the Seventh International Workshop on Business Intelligence for the Real time Enterprise (BIRTE 2013) co-located with the VLDB 2013 conference. As business analytics evolves to address new challenges such as big data with high velocity and predictive analytics, BIRTE is becoming an important venue for researchers to present recent and ongoing work in this domain. BIRTE 2013 featured an exciting technical program including two keynotes, an invited industrial talk, a panel, and a number of peer-reviewed papers from different countries in Europe, Africa, and Asia.

This issue closes with the call for papers for the Special Issue on Visionary Ideas in Data Management, to be published as the June 2015 Issue of the ACM SIGMOD Record.

On behalf of the SIGMOD Record Editorial board, I hope that you all enjoy reading the December 2014 issue of the SIGMOD Record and wish you a happy, productive year in 2015!

Your submissions to the Record are welcome via the submission site:
http://sigmod.hosting.acm.org/record

Prior to submitting, you are encouraged to read the Editorial Policy on the SIGMOD Record's Web site (http://www.sigmod.org/publications/sigmod-record/sigmod-record-editorial-policy).

Yanlei Diao

December 2014

Past SIGMOD Record Editors:

Harrison R. Morse (1969)
Daniel O'Connell (1971 – 1973)
Randall Rustin (1974-1975)
Douglas S. Kerr (1976-1978)
Thomas J. Cook (1981 – 1983)
Jon D. Clark (1984 – 1985)
Margaret H. Dunham (1986 – 1988)
Arie Segev (1989 – 1995)
Jennifer Widom (1995 – 1996)
Michael Franklin (1996 – 2000)
Ling Liu (2000 – 2004)
Mario Nascimento (2005 – 2007)
Alexandros Labrinidis (2007 – 2009)
Ioana Manolescu (2009-2013)

Foundations of Crowd Data Sourcing

Yael Amsterdamer and Tova Milo

{yaelamst, milo}@post.tau.ac.il
Tel Aviv University, Tel Aviv, Israel

ABSTRACT

Crowdsourcing techniques are very powerful when harnessed for the purpose of collecting and managing data. In order to provide sound scientific foundations for crowdsourcing and support the development of efficient crowdsourcing processes, adequate formal models must be defined. In particular, the models must formalize unique characteristics of crowd-based settings, such as the knowledge of the crowd and crowd-provided data; the interaction with crowd members; the inherent inaccuracies and disagreements in crowd answers; and evaluation metrics that capture the cost and effort of the crowd. In this paper, we review the foundational challenges in modeling crowd-based data sourcing, for its two main tasks, namely, harvesting data and processing it with the help of the crowd. For each of the two task types, we dive into the details of one foundational line of work, analyzing its model and reviewing the theoretical results established using this model, such as complexity bounds and efficient algorithms. We also overview a broader spectrum of work on crowd data sourcing, and highlight directions for further research.

1. INTRODUCTION

Crowd-based data sourcing is an emerging data procurement paradigm that engages Web users to collectively contribute and process information [8]. Well-known crowd platforms include Wikipedia,[1] websites of reviews and ratings such as IMDB,[2] and many more.

In order to work with the crowd, one has to overcome several challenges, such as dealing with users of different expertise and reliability, and whose time, memory and attention are limited; handling data that is uncertain, subjective and contradictory; and so on. Particular crowd platforms typically tackle these challenges in an ad-hoc manner, which is application-specific and rarely sharable. These challenges along with the evident potential of crowdsourcing have raised the attention of the scientific community, and called for developing sound foundations and provably efficient approaches to crowdsourcing. The present paper surveys *established theoretical foundations of crowdsourcing*, reviews a variety of approaches and results, and highlights remaining questions and directions for future research.

We start, in Section 2, by discussing an essential but challenging aspect of crowdsourcing, namely, *providing formal models for the crowd*. On the one hand, such models are necessary for studying the theoretical foundations of crowdsourcing. On the other hand, the behavior of the crowd may be very unexpected, and hard to formalize. We consider three aspects of crowdsourcing models, namely *the data model* (Section 2.1), *the interface with the crowd* (Section 2.2), and *the target function* of the crowdsourcing process (Section 2.3). For each of the three, we survey the different modeling considerations, and solutions that address them.

For example, consider a dietician that wishes to study the culinary habits of people in a certain population, e.g., combinations of food dishes that are consumed together with caffeine. The culinary habits within the population may not be recorded anywhere, and are individual, so the data must be collected by asking people about their habits. For this purpose, crowdsourcing is a notably powerful tool, as it enables a systematic exploration of a potentially huge data space (relevant combinations of food dishes) in a manner reminiscent of database exploration. Considering the underlying model for such an example, the data model must account for the unrecorded, personal habits of crowd members. One must also define the interface with the crowd, namely what types of questions are posed to crowd members (e.g., "When you eat X, how often do you also eat Y?"), what types of answers are expected (e.g., selecting a frequency for the habit in question from a drop-down list), and how the answers are analyzed by the system. The latter task is par-

[1] `http://en.wikipedia.org/`
[2] `http://www.imdb.com`

ticularly challenging, since the crowd can provide answers that are approximate, contradictory and even malicious. Last, the process of harvesting the culinary habits from the crowd must be guided by some target function. For example, since the crowd is an expensive resource, a typical target function is learning about the habits of crowd members while minimizing their effort. Additional considerations (such as the quality of the computed results) and the tradeoff between them are discussed in Section 2.3.

The crowd could be harnessed for various data-related tasks, which can be divided into two main types. First, the crowd could be engaged in *harvesting new or missing data*; and second, the crowd can *process data* that was already collected, by providing their judgments, comparing, cleaning and matching data items. Both types of tasks have been studied in previous literature, e.g., [3, 6, 7, 9, 12, 13, 14, 15, 17, 19, 24, 26, 28]. However, much of the work has been focused on the technical and practical aspects of crowdsourcing. To gain a deeper understanding of the theoretical foundations of crowdsourcing, we provide a more complete picture for two foundational research studies, one for harvesting data and one for processing it with the crowd.

In Section 3, we consider a recently developed approach for harvesting the crowd, namely, *crowd mining* [3], which focuses on identifying statistically significant patterns within the habits and preferences of the crowd. Crowd mining can be used, e.g., for identifying popular combinations of food dishes (useful to the dietician from our previous example). Crowd mining poses several theoretical challenges: first, its model must account for the individual data of the crowd which is mined in this process, as well as for a notion of *overall significant* data patterns in a given population (rather than in the habits of a single person). We outline this in Section 3.1. Second, similarly to standard data mining techniques, the space of patterns may be huge, and efficient algorithms are required for making this approach feasible. Due to the crowd involvement, the setting is quite different from that of standard data mining, and calls for the development of dedicated solutions, as described in Section 3.2.

To exemplify the second type of crowdsourced tasks, namely, data processing tasks, we consider in Section 4 the crowdsourced implementation of two classic database operators, top-k and group-by, which has been studied in [7]. For example, consider the grouping of photos of individuals by person and finding the most recent photo within each cluster, something which is easy for humans to do but difficult to evaluate by machines (assuming that person name tags and photo dates are unavailable or unreliable). One promising approach for performing such a task is with the help of the crowd. In Section 4.1, we describe the theoretical model defined in [7] for the crowd, and how the top-k and group-by operators can be evaluated with the crowd. Unlike crowd mining, where the considered data is individual, in this setting we assume that there is an (unknown) ground truth, i.e., the true top-k most recent photos of each person; however, this also means that crowd members may make mistakes, e.g., wrongly identify the most recent photo. We describe probabilistic models, which capture the likelihood of errors in crowd answers, as a function of the question difficulty. The target function is then minimizing the number of questions to the crowd, while guaranteeing a low (fixed) overall probability of error. In Section 4.2 we outline the complexity bounds of algorithms that achieve this target, and show that they are quite efficient.

We further discuss additional notable foundational works about crowd data sourcing, focused on data harvesting and processing, at Sections 3.4 and 4.3, respectively. We conclude in Section 5.

2. MODELING THE CROWD

We next discuss the challenges in providing an *adequate formal model* to crowdsourcing, which enables theoretical analysis and the development of efficient algorithms. The discussion is divided into the three aspects of crowdsourcing modeling mentioned in the Introduction, namely *data modeling*, modeling the *interface with the crowd* on top of this data model, and the *target function* of the entire process, which guides the interaction with the crowd.

2.1 Models of Data

The modeling of data in crowdsourcing leverages on existing models such as relational databases [7, 10, 16, 18, 20, 21], tree or graph-shaped data [17], RDF [2], and so on. In cases where the crowd is only employed to *process* the data (e.g., filter, group or sort), standard data models can be used as-is. The novelty here lies in cases when some of the data is harvested with the help of the crowd. One can generally distinguish between procuring two types of data: *general data* that captures "general truth" that typically resides in a standard database, e.g., the locations of places or opening hours; versus *individual data* that concerns individual people, such as their preferences or habits.

To capture harvested general data in crowdsourcing, one can use models of incomplete data, e.g., relational tables with missing values, rows and/or

columns, designated to be filled in by the crowd [20, 21]. In contrast, individual data is typically not recorded in a systematic manner, and can only be collected by posing questions to people. Such data can thus be modeled as per-crowd-member knowledge bases, which are not materialized and are accessed by restricted means of interacting with the crowd [1, 3]. See a concrete such model in Section 3.1.

2.2 Models of Crowd Interface

Interacting with the crowd can be done in various, potentially intricate ways. For instance, in Wikipedia crowd members can add, edit and delete encyclopedic entries, participate in discussions and more. However, to enable exact analysis of crowdsourcing algorithms, this interaction must be defined in a precise manner: what *types of tasks* can be submitted to the crowd, what is the *possible range of answers or actions* of the crowd in response, and how these are *interpreted* by the system. These should reflect, as accurately as possible, the expected behavior of the crowd. For example, common tasks to the crowd include classifying single items [6, 19, 23, 24], comparing sets of items [9, 7, 11, 29], and providing missing data items and values [10, 20, 21, 25] in domains where human judgment is required (e.g., ratings, recommendations, semantic analysis of images or text, etc.). The answers may be interpreted as samples of the full population's knowledge (which allows estimating this full knowledge), as votes (majority vote can be used to reconcile conflicting crowd answers), as uncertain data (see below) or as data annotated with trust levels (based on the crowd members that provided it).

In particular, *potential errors* in crowd answers must be accounted for [10, 14, 20]. This is typically done in crowdsourcing using error probability models. For example, in [7], the error model captures the increase in the error probability of increasingly difficult tasks, as described in Section 4.1. The error probability can be estimated based on preliminary tests with gold-standard data (where the ground truth is known) [6, 19] or by comparing the answers of different users to the same question [3, 14]. Error estimations for individual tasks allow estimating and adjusting the *overall* probability of error, by assigning tasks with higher uncertainty to more crowd members. See Section 4.1.

2.3 Models of the Target Function

Several factors affect the performance of crowd data sourcing processes.

- Traditional computational factors: *computational and space complexity*, *network load*, etc.
- Factors affected by the *number of tasks* posed to the crowd:
  - *Latency* – the response time of the crowd is relatively slow, and may be reduced by executing several tasks in parallel [20].
  - *Monetary cost* – crowd members may be paid for performing tasks, as in crowdsourcing platforms like Amazon Mechanical Turk.[3]
  - The *attention and amount of effort* of a crowd member may be limited. [4]
- Factors reflecting the *output quality*:
  - *Output errors* – may occur because of inaccurate crowd answers [7].
  - *Faithful representation of the trends in the target population* – relevant for the harvesting of individual data, where results are typically computed based on the answers of only a small fraction of the population [3].
  - *Coverage* – the number of collected or processed data items [20].

There exist natural tradeoffs between the aforementioned factors. E.g., computing the optimal questions to the crowd (which may be computationally expensive) may reduce the number of required questions [1]; or, executing many crowd tasks in parallel may reduce the overall latency but lead to the execution of redundant tasks [6, 20]. This means that any crowdsourcing process can only optimize some of the factors, depending on the application. Other factors may be fixed or bounded (e.g., the budget). In the sequel, we describe several target functions used in specific crowdsourcing processes.

3. HARVESTING DATA WITH THE CROWD

As promised in the Introduction, in this section we will exemplify the use of crowdsourcing for harvesting data from the crowd, by diving into the theoretical foundations of a particular paradigm, namely, **crowd mining**, which have been studied in [1, 2, 3]. We start by providing a general motivation for crowd mining, then detail the underlying formal model established in [1, 2, 3] and overview some of the main theoretical results. Throughout the section, we use the culinary preferences example from the Introduction as a running example, but note that crowd mining applies to many other real-life scenarios that involve collecting data about the habits and preferences of the crowd [3].

Classic data mining algorithms discover interesting patterns in large data sets. Such algorithms typically assume that the transactions to be mined (sets of co-occurring data items) are stored in a database. In contrast, individual data is typically

[3] `https://www.mturk.com/`

not recorded in a systematic manner for large populations, and thus crowdsourcing may be required for mining such data.

The individual knowledge of a people can be modeled per-person databases, accessed by posing questions to the relevant person. These databases cannot be materialized, since asking a person to recall and report every piece of knowledge is usually impossible. Instead, the model of [3] relies on the common ability of people to recall information about their habits in the form of summaries, as shown by social studies [5]. For instance, people can report how often they eat a specific combination of food dishes. Crowd mining methods thus ask crowd members to specify their habits and/or provide the frequency at which they engage in these habits. The answers of several users are aggregated to estimate the overall significance of a habit (data pattern) in the population.

In [3], a framework for estimating the confidence in the habit significance is described, and is employed for deciding which question to ask the crowd next (see a review below). In [1], this framework is extended by leveraging semantic connections between data items in order to efficiently explore the space of data patterns and identify the significant ones. E.g., if it is known that few people in a certain population drink coffee, and that espresso is a type of coffee, it is useless to ask people about their espresso drinking habits. Finally, in [2], the crowd mining model is extended to RDF-style facts[4] rather than sets of items, and a novel, rich query language allows specifying complex data patterns of interest. The sound theoretical results of [1] are proved to extend to the more expressive setting of [2].

3.1 Formal Model

We next describe a combined formal model for the crowd mining setting of [1, 3]. For simplicity, we discuss the most basic setting, and mention how it can be extended later, in Section 3.3.

Data model. Let $\mathcal{I} = \{i_1, i_2, i_3, \dots\}$ be a finite set of item names. Define a *database* D as a finite bag (multiset) of *transactions* over $\mathcal{I}$, s.t. each transaction $T \in D$ represents an occasion, e.g., a meal. We start with a simple model where every T contains an itemset $A \subseteq \mathcal{I}$, reflecting, e.g., the set of food dishes consumed in a particular meal. Let $\mathcal{U}$ be a set of users. Every $u \in \mathcal{U}$ is associated with a *personal database* D_u containing the transactions of u (e.g., all the meals in u's history). $|D_u|$ denotes the number of transactions in D_u. The frequency or *support* of an itemset $A \subseteq \mathcal{I}$ in D_u is $\text{supp}_u(A) := |\{T \in D_u \mid A \subseteq T\}| / |D_u|$. This individual significance measure will be aggregated to identify the overall frequent itemsets in the population. For example, in the domain of culinary habits, $\mathcal{I}$ may consist of different food dishes, drinks, etc. A transaction $T \in D_u$ will contain all the items in $\mathcal{I}$ consumed by u in a particular meal. If, e.g., the set {coffee, fruits, yogurt} is frequent, it means that these food and drink items form a frequently consumed combination.

As noted in [22], there may be dependencies between *itemsets* resulting from semantic relations between *items*. For instance, the itemset {chocolate, coffee} is semantically implied by any transaction containing {chocolate, espresso}, since espresso is a (type of) coffee. Such semantic dependencies can be naturally captured by a *taxonomy* [22]. Formally, we define a taxonomy Ψ as a partial order over $\mathcal{I}$, such that $i \leq i'$ indicates that item i' is more specific than i (any i' is also an i).

Based on $\leq$, the semantic relationship between items, we can define a corresponding order relation on itemsets.[5] For itemsets A, B we define $A \leq B$ iff every item in A is implied by some item in B. We call the obtained structure the *itemset taxonomy* and denote it by $\text{I}(\Psi)$. $\text{I}(\Psi)$ is then used to extend the definition of the support of an itemset A to $\text{supp}_u(A) := |\{T \in D_u \mid A \leq T\}| / |D_u|$, i.e., the fraction of transactions that *semantically imply* A.

Crowd interface. In our crowd-based setting, the personal database D_u is not materialized and only models the knowledge of u, so we can only access D_u by asking u questions. As shown in [3], one can ask people for summaries of their personal knowledge, and then interpret them as data patterns – itemset frequencies in our case. We thus abstractly model two types of *crowd questions*, as follows.

- **Closed question.** Parameterized by an itemset $A \subseteq \mathcal{I}$ and asks a user u for $\text{supp}_{D_u}(A)$.
- **Open questions.** Asks a user u to provide some itemset $A \subseteq \mathcal{I}$ along with $\text{supp}_{D_u}(A)$.

Intuitively, closed questions are useful for computing the significance of a particular itemset, whereas open questions are useful for discovering previously unknown items (if the items domain is not given in advance) and for quickly finding itemsets that represent prominent data patterns (which are more likely to be spontaneously recalled by users).

[4] `http://www.w3.org/standards/techs/rdf`

[5] Some itemsets that are semantically equivalent are identified by this relation, e.g., {coffee, espresso} is represented by the equivalent, more concise {espresso} (since drinking espresso is a particular case of drinking coffee), see [1] for full details.

		With respect to the input	With respect to the input and output
Crowd Complexity	**Lower**	$\Omega(\log \lvert \mathrm{S}(\Psi) \rvert)$	$\Omega(\lvert mfi \rvert + \lvert mii \rvert)$
	Upper	$\mathrm{O}(\log \lvert \mathrm{S}(\Psi) \rvert)$	$\mathrm{O}(\lvert \Psi \rvert \cdot (\lvert mfi \rvert + \lvert mii \rvert))$
Comput. Complexity	**Lower**	$\Omega(\log \lvert \mathrm{S}(\Psi) \rvert)$	EQ-hard
	Upper	$\mathrm{O}\Big(\lvert \mathrm{I}(\Psi) \rvert \cdot \Big(\lvert \Psi \rvert^2 + \lvert \mathrm{I}(\Psi) \rvert\Big)\Big)$	$\mathrm{O}\Big(\lvert \mathrm{I}(\Psi) \rvert \cdot \Big(\lvert \Psi \rvert^2 + \lvert mfi \rvert + \lvert mii \rvert\Big)\Big)$

Table 1: Summary of crowd mining complexity results, where $|\mathrm{I}(\Psi)| \leq 2^{\mathrm{O}(|\Psi|)}$ and $|\mathrm{S}(\Psi)| \leq 2^{\mathrm{O}(|\mathrm{I}(\Psi)|)}$

Target function. To compute the *overall significance* of an itemset A, we need to *aggregate* the support of A in the databases of the users in U, and apply a *predicate* for deciding whether the result is significant or not. In [3], *average* is used for aggregation, and the itemset A is considered significant (frequent) if its support exceeds a predefined threshold Θ.

In practice, it is impossible to obtain the answers of all the users about a certain rule. Thus, in [3], a sample-based empirical estimation is employed by posing questions about each data pattern to a random (uniform) sample of the crowd members. In a nutshell, given a small set of user answers regarding a particular itemset, the technique of [3] estimates the *unknown distribution* of the mean of these answers (the aggregated result). As more answers are obtained from the crowd, this distribution converges to the true mean in the entire population. The decision whether an itemset is significant is done based on the probability that the mean support of this itemset exceeds the threshold Θ.

Based on the above setting, the target function of crowd mining can be defined in different ways. In [3], a greedy algorithm is considered, whose target is to choose the next crowd question that minimizes the *overall expected uncertainty* – the single optimization factor. The uncertainty is defined based on estimating, per data pattern, the probability that it was wrongly classified as (in)significant. An alternative target function considered in [1] aims to estimate the significance of *all the itemsets*, while minimizing first the number of questions posed to the crowd (termed *crowd complexity*), and then the computational complexity of selecting these questions. The uncertainty is assumed to be controlled by asking a sufficient number of users about each itemset. I.e., the error is bounded while the crowd and computational complexities are optimized.

3.2 Theoretical Results

The model for crowd mining sets the formal foundations for the development of provably efficient crowdsourcing algorithms, as well as studying their complexity bounds. We next overview such results for the basic model described above, which are fundamental and can be adapted for more complex setting (see Section 3.3). Formally, we define the problem of `CrowdMine` as follows: given a domain of items $\mathcal{I}$, a taxonomy Ψ over its items, find the frequent itemsets in the induced taxonomy of itemsets $\mathrm{I}(\Psi)$, by posing closed questions to the crowd. (Open questions are further considered in [3, 2], but the theoretical results below only apply to close questions.) The following theorem summarizes the main complexity results for `CrowdMine` established in [1].

THEOREM 3.1 ([1]). *The complexity bounds of solving* `CrowdMine` *are stated in Table 1.*

In the first column of Table 1, complexity bounds are given in terms the input taxonomy Ψ. In contrast, in the second column, complexity bounds are given in terms of both the input and the output of the mining process, namely, the number of maximal (most specific) frequent itemsets (MFIs) and minimal (most general) infrequent itemsets (MIIs). Intuitively, the MFIs and MIIs are alternative concise descriptions of the frequent itemsets, and thus capture the output of the mining process.

The first row of Table 1 presents *crowd complexity* results, defined as the number of distinct itemsets the crowd is asked about (assuming questions are posed to a sufficient number of users to determine the itemset frequency). Given a taxonomy Ψ, `CrowdMine` has a tight bound logarithmic in $|\mathrm{S}(\Psi)|$, the number of possible Boolean frequency functions, which depends on Ψ. As reflected in the inequalities at the bottom of Table 1, $\log |\mathrm{S}(\Psi)|$ is at most exponential in $|\Psi|$. When the output is considered, the lower complexity bound is the sum of the numbers of MFIs and MIIs, and the upper bound adds the taxonomy size as a multiplicative factor (i.e., its complexity nearly but not exactly achieves the lower bound).

Algorithm
Input: R: the most general data pattern
Output: The set of MFIs M
1 Add R to an empty priority queue Q;
2 $M \leftarrow \emptyset$;
3 **while** Q contains unclassified data patterns **do**
4 $A \leftarrow$ the minimal pattern in Q ;
5 **if** ask(A) **then**
6 **while** exists unclassified B s.t. $A \lessdot B$ **do**
7 **if** ask(B) **then** $A \leftarrow B$;
8 add A to M;
9 **return** M;

Algorithm 1: Crowd mining algorithm

This is proved by providing a constructive algorithm, outlined in the sequel.

The second row of the table, presents *computational complexity* bounds, of performing an optimal selection of the itemsets the crowd will be asked about throughout the mining process. The crowd complexity lower bound is trivially a lower bound of computational complexity, but w.r.t. the output a stronger hardness result is obtained by showing that the problem is EQ-hard in the taxonomy size and in the numbers of MFIs and MIIs. EQ is a basic problem in Boolean function learning, not known to be solvable in PTIME [1]. The upper bounds in the bottom row are achieved by the same algorithm that achieves the crowd complexity upper bound, and are polynomial in $|\mathrm{I}(\Psi)|$, and at most exponential in $|\Psi|$. See [1] for the proofs of these theoretical results.

The algorithm used in all the constructive[6] upper bound proofs in [1] is presented next in a slightly simplified manner in Algorithm 1, based on its extension from [2]. Although the algorithm is simple, it is both complexity-wise efficient, useful in practice, and can be adapted for more complex settings [2].

Algorithm 1 maintains a priority queue Q where itemsets are ordered from the most general to the most specific. Q is initialized with the single most general pattern. Iteratively, the algorithm pops out an unclassified data pattern A from Q, which is not known to be (in)significant yet. To compute whether the pattern is significant, the algorithm uses the function ask($\cdot$), which abstracts, for simplicity, the process of posing questions about A to several crowd members. The algorithm then explores subsequent patterns in the partial order (using the $\lessdot$ relation), searching for patterns that are both more specific and significant, until it discovers an MFI. Every call to ask($\cdot$) can classify multiple patterns, using *semantic inference*: if $A \leq B$ and B is significant, so is A; and vice versa.

The number of MFIs and MIIs is typically much smaller than the number of itemsets [1]. Recall that the crowd complexity of Algorithm 1 described above is $\mathrm{O}(|\Psi| \cdot (|mfi| + |mii|))$. Intuitively, the inner loop of the algorithm can check at most $|\Psi|$ itemsets, since we can attempt add each item to the current itemset at most once (and if this results in an insignificant itemset B, all the more specific itemsets C s.t. $B \leq C$ are inferred to be insignificant). Since each loop identifies an MFI or an MII (if the first itemset popped from Q is insignificant), we obtain the crowd complexity bound mentioned above.

[6]The exception is the upper crowd complexity w.r.t. the input size, whose proof is a non-constructive one.

3.3 Extensions

The simple model of mining itemsets cannot capture all types of interesting patterns that one might want to mine. E.g., it can express sets of co-occurring items, but not more complex relationships between them. This model is thus extended in [3] to mining *association rules*: given two itemsets A and B, $A \rightarrow B$ is an association rule denoting an implication between A and B, which can be viewed as a likelihood of transactions which contain A to also contain B. For example, $\{\mathsf{coffee},\mathsf{fruits}\} \rightarrow \{\mathsf{yogurt}\}$ can signify that whenever people have coffee and fruits, they are likely to also have yogurt. In [2], even more complex data patterns are considered, namely sets and association rules of RDF-style *facts*. Given a vocabulary of items $\mathcal{I} = \mathcal{E} \cup \mathcal{R}$, a fact $f \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ denotes a relationship $r \in \mathcal{R}$ between two items $e_1, e_2 \in \mathcal{E}$, e.g., the fact Coffee drinkAt Starbucks denotes a habit of drinking coffee in Starbucks. The semantic order relation between itemsets is extended in [2] to support sets of facts.

Algorithm 1 supports identifying MFIs within the entire space of itemsets. However, people often may be interested only in specific patterns. To restrict the search, the work of [2] introduces a query language called `OASSIS-QL`, which allows to specifying subset of data patterns that are of interest to the user, and mining only them from the crowd. Extending our example from the beginning of the section, the dietician may only be interested in studying culinary preferences with respect to dishes with caffeine, or meals eaten at restaurants in a certain region. Using a syntax based on SPARQL, the RDF query language, `OASSIS-QL` allows selecting only the relevant food fishes/ restaurants from a knowledge base. The selected items are then used to generate rel-

evant habits on which questions are posed to the crowd. It is proved in [2] that given a query, a notion equivalent to MFI and MII can be defined over the patterns that match it; then, Algorithm 1 is adapted to support finding such MFIs and MIIs, which gives a complexity bound similar to the one obtained in [1] for itemsets and without a query.

3.4 Harvesting General Data

The model and algorithms described above only deal with harvesting individual data from the crowd. Another line of work in crowdsourcing considers harvesting general data from the crowd. For instance, some recent work (e.g., [10, 16, 20, 21]) suggests the construction of declarative frameworks, which outsource the harvesting of certain missing values or missing tuples to the crowd. While most of this work focuses on the practical and technical aspects of the problem, we mention here two example works that focus on the theoretical aspects of general data collection with the crowd.

The first is [4], which makes the closed-world assumption that there is a known set of missing values. Each value could be fetched by posing a specific question to a sufficient (fixed) number of crowd members. The paper considers several target functions, including minimizing quality-related factors such as the max or the sum of uncertainties over all the values, while fixing cost-related parameters such as the overall number of questions asked, the maximal number of questions per user, etc. The results in [4] establish the complexity bounds of this problem and include constructive, efficient algorithms where possible. In particular, all the considered variants of the problem are proven to be in PTIME, except for optimizing the sum of uncertainties given a bounded number of questions per-user, or when the users are asked groups of overlapping questions. These exceptions are proved NP-hard.

The second work we mention is [25], which studies the harvesting of data under an open-world assumption. Many answers for a given question are collected from the crowd, e.g., the crowd can be asked to list ice-cream flavors. The key observation is that the more answers are collected, the next answer is less likely to provide a new value not obtained before. For example, after the prominent ice-cream flavors has been collected (e.g., vanilla, chocolate) a crowd member is less likely to spontaneously choose a flavor that is not in the list. To address this phenomenon, statistical tools are developed in [25], for estimating the future rate of incoming new values based on the rate observed thus far. These tools support various target functions that balance the cost (number of questions) versus coverage (number of unique values obtained) tradeoff.

4. DATA PROCESSING WITH THE CROWD

In addition to harvesting data from the crowd, the crowd has been proved to be very effective in tasks of processing data, such as cleaning, sorting and matching data items (e.g., [9, 11, 15, 23, 24, 27, 29]). In particular, some recent work (e.g., [6, 7, 12, 14, 15, 17, 19, 26, 28]), including some of the declarative crowdsourcing frameworks mentioned before [10, 16, 20], consider the execution of common query operators such as filter, join, count and max. As an example, we elaborate, in this section, on the theoretical foundations of computing **top-k and group-by queries**, studied in [7].

Suppose we have a database of unlabeled photos, `PhotoDB`, and we are interested in executing the following query over it.

```
SELECT Most-recent(photo)
FROM PhotoDB
GROUP BY Person(photo)
```

`PhotoDB` contains only photos of a single person (whose face can be recognized), e.g., Alice in her office or Bob in front of the Louvre, but not of Alice and Bob together. We now wish to: (i) group (cluster) the photos by the person they represent; and (ii) find the most recent photo – or the k most recent photos – of each person (max/top-k).

The query above includes two user-defined functions: `Person` clusters photos of the same person, and `Most-recent` selects the most recent photo within each cluster. Note that the most recent photo is also the one in which the photographed person is the oldest. While the `Person` function might be performed by face recognition software, it is slow and costly. Furthermore, the results may not be impressive for a time span of 20 years during which the person ages from babyhood to being an adult. As for ordering the photos by date, we assume there is no trustworthy date of when the photo was taken. This may be the case due to untuned dates in the camera, scanned photos, etc. Thus, we ask crowd members to identify the person in the photo and compare the photo dates based on the person's age and perhaps other cues.

4.1 Formal Model

The data model in this setting is a standard one, namely, a repository of photos. The user-defined functions we wish to execute with the help of the crowd may be viewed as fetching meta-data about the photos. We next detail the crowd interface

Max/Top-k		Clustering		Correlated Clustering
$\Theta\big(n \log \frac{1}{\delta}\big)$	(constant error)	$\Omega(nJ)$		$\mathrm{O}\big((n \log(\alpha J) + \alpha J) \log \frac{n}{\delta}\big)$
$n + \mathrm{o}\big(\frac{n}{\delta} \log \frac{1}{\delta}\big)$	(variable error in general)	$\mathrm{O}\big(nJ \log \frac{n}{\delta}\big)$	(in general)	
$n + \mathrm{O}\Big(\frac{\log \log n}{\delta^2} \log \frac{1}{\delta}\Big)$	(variable error, $f(\Delta) = \Omega(\Delta)$)	$\mathrm{O}(nJ)$	(when $\varepsilon = \frac{1}{2}$)	
$n + \mathrm{O}\big(\log^2 \frac{1}{\delta}\big)$	(variable error, $f(\Delta) = 2^{\Delta}$)			

Table 2: Summary of top-k and group by complexity results; k is constant; n is the database size; δ is the maximal error; $f(\Delta)$ is the error as a function of the distance between items Δ; J is the number of clusters; $\frac{1}{2} - \varepsilon$ is the user error; α is a correlation factor.

model assumed in this setting, along with the model of crowd errors.

Crowd interface. Two types of questions are used to respectively group and order data items (photos).

- **Type question.** Given two data items i, i' return `TRUE` iff they belong to the same type, e.g., the two photos are of the same person.
- **Value questions.** Given two data items i, i' return `TRUE` iff $i < i'$ by the order relation over items, e.g., the photo i was taken before i'.

Given a perfect oracle that answers these questions, it is clearly possible to group the data items and partially sort them to find the top-k. However, the crowd may make mistakes, i.e. they may not correctly identify two pictures as being of the same person (type error) or of one picture of a person being more recent than another picture of that same person (value error). Two models of error are considered in [7], as follows.

- **Constant error model:** Each type or value question is answered correctly by the crowd with a constant probability $> \frac{1}{2}$.
- **Variable error model:** Models an increase in error probability for items that are closer in the considered ordering.

While the constant error model is more standard (and used in used in previous crowdsourcing work, e.g., [6, 19]), the variable model is novel. It captures the intuition that the closer items are in the order relation, the harder it is for a crowd member to correctly order them. It is much easier, e.g., to compare the dates of two photos of the same person with 10 years apart than photos with one week apart, and the error probability increases accordingly.

Target function. Using either of the error models mentioned above, it is impossible to guarantee perfect clustering or top-k selection. Hence, the techniques described in [7] focus on bounding the total error probability: given an arbitrarily small constant $\delta \in (0, \frac{1}{2})$, the techniques compute the correct answer with probability $> 1 - \delta$. Within this bound, the techniques aim to perform the grouping and/or top-k selection tasks while minimizing the number of questions posed to the crowd.

4.2 Theoretical Results

Let us formally define the problems of computing top-k and group by queries with the crowd: let n be the number of items in the database; k a constant; and $\delta > 0$ the maximal allowed overall probability of error. `TOPk` is the problem of computing the top-k items in the database, `CLUSTER` is the problem of clustering the n items, and `CCLUSTER` is the problem of clustering when the clusters are correlated with an order over the data items. For the three problems, the correct answer must be computed in probability $\geq 1 - \delta$, and the complexity is measured by the number of questions posed to the crowd. The next theorem summarizes the main results for these problems, achieved in [7].

THEOREM 4.1 ([7]). *The complexity bounds of solving `TOPk`, `CLUSTER` and `CCLUSTER` are stated in Table 2.*

We next briefly explain these results.

`TOPk`. It is proved in [7] that, when each value question is answered correctly with probability $\frac{1}{2} + \varepsilon$ for a constant ε, the maximum can be computed with probability $\geq 1 - \delta$ in time $\mathrm{O}\big(n \log \frac{1}{\delta}\big)$; they also show that this bound is tight. In the novel variable error model a much better upper bound can be obtained: suppose the two elements being compared by a value question are Δ apart in the sorted order. Then the probability of error is $\leq \frac{1}{f(\Delta)}$ for a monotone, non-negative error function f, i.e., the error in the answer decreases when the distance Δ between the elements increases. For `TOPk` (with a constant k), $n + \mathrm{o}(n)$ value questions are sufficient given any strictly monotone error function f, where

$o(n)$ denotes a function that is strictly asymptotically smaller than n.

CLUSTER. For the general clustering problem using the fixed cost model, a lower bound of $\Omega(nJ)$ is achieved for discovering J clusters. For the upper bound, a simple algorithm that compares $O(nJ)$ pairs of elements by type questions proves that the lower bound is tight when $\varepsilon = \frac{1}{2}$; if the answers to the type questions are erroneous ($\varepsilon < \frac{1}{2}$), the number of questions increases by a factor of $O(\log n)$.

CCLUSTER. Consider the case when item types are correlated with an order over item values. E.g., suppose we have a database of hotels in a city, to be clustered by districts. Then there may be a high correlation between the district and the hotel rating. Formally, let $\alpha \in [1, n-1]$ be the correlation factor between the value and type (order and clustering), defined as follows: sort the items according to their value; the *distance* between elements x_i and x_j of type T is $|\{x_l | x_i < x_l <, x_l \text{ is not of type } T\}|$. α is the maximal such distance $+1$. When α is small the correlation can be leveraged to reduce the clustering complexity [7].

4.3 Additional Crowdsourced Operators

To complete the picture, we note a few more example works about other data processing operators. The problem of discovering the *maximum element* in a set of data items, which is a sub-problem of [7] (top-1), was also studied in [12]. However, instead of finding the maximum element exactly, [12] focuses on the judgment problem (which element has the maximum likelihood of being the maximum element) and the next vote problem (which future comparisons will be most effective). Finding the exact solution to these problems was shown to be hard, and instead, efficient heuristics were proposed.

The crowd sourcing of *filter* or *selection* operator, namely, using humans to provide a binary (or, in general, n-ary) evaluation of data items, was studied in [19], and later extended in [18]. Such an operator can be used, e.g., to filter images in a database, by asking crowd members to evaluate each image. The main challenges addressed in this work include the modeling the error of the crowd, and computing a strategy for deciding whether to pose some question to more people or make a decision. The model of [19] assumes that the error probabilities are fixed and known, which is later relaxed in [18] to support per-worker and per-question error probabilities. The complexity bounds of the problem were studied in [18, 19] for various target functions, which balance cost and accuracy in different ways.

We also note the work of [17] about crowd-assisted *searching of elements* within a directed acyclic graph, by asking reachability questions. For example, this operator can be used for classifying documents within a hierarchy of categories, by asking "Does document X belong to category Y?", which is interpreted as "Is the most specific category of X reachable from Y?", assuming that directed edges signify category subsumption. Two target functions for this problem were considered in [17], of either minimizing the number of questions while obtaining full coverage, or maximizing the accuracy within a fixed budget of questions. The complexity bounds of computing which questions to ask were studied for different properties of the graph structure, and efficient algorithms were proposed, where possible.

5. CONCLUSION

In this paper, we discussed the foundations of crowd data sourcing, focusing on its two main uses, namely, harvesting data and processing it with the crowd. We have highlighted the various challenges in providing adequate formal models for the crowd, and the general components of existing solutions that alleviate these challenges. We have then reviewed in more detail a few particular examples for crowdsourcing works, in light of their specific challenges, solutions and established results. We note that many other related challenges were considered by literature on crowdsourcing, which are out of the scope of this paper. For instance, these include the construction of data queries with the help of the crowd; the management of reward offered to crowd members; the assignment of tasks based on crowd member expertise and availability; and so on.

The survey of crowdsourcing works reveals some interesting challenges for future work. In particular, harvesting both individual and general data together has not been considered, and may be beneficial, e.g., for dynamically refining a general knowledge base according to harvested individual data. In addition, crowdsourcing problems are typically studied with respect to a predefined target function. A more generic solution could employ a parameterizable target function, to support customizing the balance between different cost and quality factors. Other challenges include dealing with privacy issues; the selection of crowd members according to their profiles; the design of user interface; and the maintenance of crowd-provided data, to account for trust, staleness, etc. Finally, richer crowd and error models that capture, more fully, different aspects of user behavior are also an intriguing research direction.

Acknowledgments. We are very grateful to Antoine Amarilli, Susan B Davidson, Yael Grossman, Sanjeev Khanna, Slava Novgorodov, Sudeepa Roy, Pierre Senellart and Amit Somech, our collaborators on the main work surveyed in this paper. We also thank the anonymous reviewer for useful comments.

This work has been partially funded by the European Research Council under the FP7, ERC grant MoDaS, agreement 291071, and by the Israel Ministry of Science.

6. REFERENCES

[1] A. Amarilli, Y. Amsterdamer, and T. Milo. On the complexity of mining itemsets from the crowd using taxonomies. In *ICDT*, 2014.

[2] Y. Amsterdamer, S. B. Davidson, T. Milo, S. Novgorodov, and A. Somech. OASSIS: query driven crowd mining. In *SIGMOD*, 2014.

[3] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart. Crowd mining. In *SIGMOD*, 2013.

[4] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W.-C. Tan. Asking the right questions in crowd data sourcing. In *ICDE*, 2012.

[5] N. Bradburn, L. Rips, and S. Shevell. Answering autobiographical questions: the impact of memory and inference on surveys. *Science*, 236(4798), 1987.

[6] A. Das Sarma, A. G. Parameswaran, H. Garcia-Molina, and A. Y. Halevy. Crowd-powered find algorithms. In *ICDE*, 2014.

[7] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In *ICDT*, 2013.

[8] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the World-Wide Web. *Commun. ACM*, 54(4), 2011.

[9] J. Fan, M. Lu, B. C. Ooi, W. Tan, and M. Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *ICDE*, 2014.

[10] M. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: answering queries with crowdsourcing. In *SIGMOD*, 2011.

[11] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*, 2014.

[12] S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD*, 2012.

[13] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. CDAS: A crowdsourcing data analytics system. *PVLDB*, 5(10), 2012.

[14] A. Marcus, D. Karger, S. Madden, R. Miller, and S. Oh. Counting with the crowd. In *VLDB*, 2012.

[15] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1), 2011.

[16] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, 2011.

[17] A. Parameswaran, A. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *PVLDB*, 4(5), 2011.

[18] A. G. Parameswaran, S. Boyd, H. Garcia-Molina, A. Gupta, N. Polyzotis, and J. Widom. Optimal crowd-powered rating and filtering algorithms. *PVLDB*, 7(9), 2014.

[19] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. CrowdScreen: algorithms for filtering data with humans. In *SIGMOD*, 2012.

[20] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM*, 2012.

[21] H. Park and J. Widom. CrowdFill: collecting structured data from the crowd. In *SIGMOD*, 2014.

[22] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB*, 1995.

[23] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. Data curation at scale: The Data Tamer system. In *CIDR*, 2013.

[24] C. Sun, N. Rampalli, F. Yang, and A. Doan. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *PVLDB*, 7(13), 2014.

[25] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar. Crowdsourced enumeration queries. In *ICDE*, 2013.

[26] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis. Max algorithms in crowdsourcing environments. In *WWW*, 2012.

[27] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. CrowdER: crowdsourcing entity resolution. *PVLDB*, 5(11), 2012.

[28] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*, 2013.

[29] C. Zhang, L. Chen, H. V. Jagadish, and C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *PVLDB*, 6(9), 2013.

Towards hardware-driven design of low-energy algorithms for data analysis

Indre Zliobaite, Jaakko Hollmen,
Aalto university and HIIT
Espoo, Finland
firstname.lastname@aalto.fi

Lauri Koskinen, and Jukka Teittinen
Technology Reserch Center, University of Turku
Turku, Finland
firstname.lastname@utu.fi

ABSTRACT

In the era of "big" data, data analysis algorithms need to be efficient. Traditionally researchers would tackle this problem by considering "small" data algorithms, and investigating how to make them computationally more efficient for big data applications. The main means to achieve computational efficiency would be to revise the necessity and order of subroutines, or to approximate calculations. This paper presents a viewpoint that in order to be able to cope with the new challenges of the growing digital universe, research needs to take a combined view towards data analysis algorithm design and hardware design, and discusses a potential research direction in taking an intreated approach in terms of algorithm design and hardware design. Analyzing how data mining algorithms operate at the elementary operations level can help do design more specialized and dedicated hardware, that, for instance, would be more energy efficient. In turn, understanding hardware design can help to develop more effective algorithms.

1. INTRODUCTION

More and more data is being generated every day by people, enterprises and smart devices. It is estimated [5] that the digital universe is growing by 40% a year, and by 2020 the data we create and copy annually will reach 44 zettabytes, or 44 trillion gigabytes. While currently less than 5% of the information in the digital universe is actually analyzed, it is estimated that at least 20% would be a candidate for analysis. Hence, there are more opportunities, but at the same time, more challenges for data analysis, and not only human resources, but also new technologies are needed to make the best use of these opportunities.

Data generated by smart devices is particularly interesting, since it presents opportunities for analysis that did not exist a few years ago. Currently data from embedded systems (Internet of Things) accounts for only about 10% of all the analyzable data, it is estimated [5] that by 2020 data from embedded systems will account for half of all the analyzable data. Such devices typically operate using autonomous power sources, therefore, research needs to anticipate how to process such data in the most energy efficient way.

This presents an interesting dilemma from the data mining perspective. Ideally, all of the data gathered in the wireless sensors would be tranferred wirelessly to be centrally processed. And while the energy consumption of data processing continually falls with Moore's Law [7, 6], the energy consumption of transmitting a bit across a given distance does not follow the Law as advantageously as the digital processing. Therefore, the energy cost of wireless transmission will proportionally grow when compared to digital processing. This means we will have to do more and more data analysis on smart devices, instead of sending it around.

Due to power constraints, it appears that the current general processor multi-core model is unlikely to scale beyond a limited number of cores; it will no longer be possible to use all cores simultaneously (a notion called Dark Silicon by ARM). This will result in heterogeneous multi-cores where only a few accelerators are used at any given time. An example can be seen in current mobile phone application processors which consist of many specialized units, for instance graphics processing accelerators, radio baseband accelerators etc. For example, Qualcomm's new Snapdragon S4, marketed as a dual-core processor, is actually a 10-core unit with a GPU, several DSPs, modem, etc. Processors will further develop into consisting of more subsystems and will therefore be heterogeneous units specialized for particular purposes.

In this article we present a viewpoint that in order to be able to cope with the new challenges of the growing digital universe, research needs to take a combined view towards data analysis algorithm design and hardware design. Analyzing how data mining algorithms operate at the elementary op-

erations level can help do design more specialized and dedicated hardware, that, for instance, would be more energy efficient. In turn, understanding hardware design can help to develop more effective algorithms.

The remainder of the paper is organized as follows. Section 2 overviews the challenges and desired properties from the hardware side. Section 3 overviews the challenges and desired properties from the algorithmic side. Section 4 analyses the existing most popular data mining/machine learning algorithms in terms of elementary operations, and relates algorithmic and hardware side. Section 5 discusses open research directions in energy efficient data mining.

2. OPPORTUNITIES AND CHALLENGES FROM THE HARDWARE PERSPECTIVE

Moore's Law has promised doubling of processing power every 18 months due to the scaling of CMOS[1] transistors, which are at the heart of all processors. While the scaling of transistors still continues, extraneous circumstances have made achieving the higher processing power increasingly difficult. About ten years ago, UC Berkeley's David Patterson defined the "Three Walls". While these design complexities (aka. "Walls") were known previously, Patterson combined them in a forward-looking equation:

Power Wall+Memory Wall+ILP Wall = Brick Wall.

These Three Walls could be shortly defined as follows:

Power Wall
If processor speed (clock frequency) would follow Moore's Law, the ensuing heat generated by the power consumption would destroy the processor. As the processor clock frequency is linearly (and superlinearly in extreme cases) related to power consumption, this can be seen in the plateauing of processor clock frequencies to c. 3-5 GHz.

Memory Wall
Ideally, all of the data to be processed would be stored adjacent to the processing element. However, with current memory technology, only small amounts can be stored beside the processor (e.g. L1 cache) and the ensuing data transfer delay stalls the processing. This can be seen in hiding memory latency with increasing on-chip cache sizes and levels and also the increasing amount of concurrently processed threads.

Instruction-Level Parallelism Wall (ILP)
The single-thread performance has been traditionally increased by parallelizing the sequential part of computer programs. This was achieved with better compilers or more complex architectures (out-of-order instruction processing, branch speculation, VLIW[2]).

To compound the problem, tackling one wall will make the other two worse.

The ultimate solution to break the Brick Wall is moving away CMOS technology to a novel technology. As such a mature technology does not yet exist, other tricks have been devised to mitigate the problem and lower the Wall. In the early 2000s processors moved from single core to multi core, but this made the Memory Wall worse. Cache coherence reduces the Memory Wall, but aggravates the Power Wall. The Power Wall currently also manifests itself in cloud computing via the electricity bill.

As smaller transistors allow cramming more cores into a single chip, Dark silicon, described in Section 1 is truly the main way to lower the Power Wall. However, two other problems ensue: the Programmer Wall and the Communication Wall.

The Programmer Wall basically relates to abstracting the heterogeneous nature of Dark Silicon as algorithms and software are not ready for computing on thousands of processors.

The Communication Wall Communication here means either moving data between levels of a memory hierarchy (sequential case), over a network connecting processors (parallel case), or wirelessly between processing nodes (sensor network case).

While creating an efficient programming model and associated compilers for Dark Silicon is out of scope for this discussion, the Communication Wall can already be tackled at the algorithm level (see e.g. [2]). Previously, the cost of communicating was divided into bandwidth and latency. However, the energy cost of communication is proportionally growing when compared to computation. As mentioned previously, due to the physics of electromagnetic signalling, wireless communication is the extreme

[1]Complementary metal oxide semiconductor (CMOS) is a technology for constructing integrated circuits.

[2]Very long instruction word (VLIW) is a processor architecture designed to take advantage of instruction level parallelism.

case. But even on-chip, the delay of wiring grows quadratically with scaling and as a result increasing amount of energy will have to be used to achieve sufficient bandwidth and delay.

To summarize, the algorithm designer should think of specialized functions that can be turned into Dark Silicon processing elements and minimize communication (memory accesses and data transfer between cores). When designing algorithms, or choosing existing algorithms for implementation, data analysis should be aware of energy costs of elementary operations at different hardware designs, and much energy which operation consumes. That would allow to prioritize. Algorithm and elementary function complexity and algorithm execution time are rudimentary estimates on energy consumption, which, in some cases, can be mitigated with Dark Silicon. Amount of memory required by the algorithm should also be minimized. Communication can be minimized by, for example, using small data sets that fit in a processor cache (and thereby avoid extra communication with the main memory) or using hierarchical algorithms (where some levels of the hierarchy can be performed in wireless nodes).

3. LOW-ENERGY ALGORITHMS

All data analysis algorithms can be broadly categorized as descriptive or predictive modeling.

Descriptive modelling aims to extract information from large amount of data and summarize it into a model. The main goal is to describe a dataset at hand, and the main result of an algorithm is data summary (e.g. clustering traces of rental bicycles into groups in order to develop new bicycle roads).

Predictive modelling aims to extract summarizing information from large amount data into a model as well, however, the main goal is not to describe the data at hand, but rather to be able to generalize to new unseen data (e.g. recognizing the model of transportation based on accelerometer data captured by a mobile device).

While in descriptive modeling the modeling algorithms need to be as efficient as possible, in predictive modeling there needs to be a tradeoff between efficient modeling and efficient generalization. For example, producing a regression model requires much more computation resources than later applying the model to new data. On the other hand, for instance, k-nearest neighbour algorithm would require much more computing resources at the application stage.

There are many strands of research related to energy-efficiency, rather scattered across different research areas. The authors in [15] insist that a paradigm shift towards energy-efficiency is needed, if we want to use thousands of battery-powered low-cost sensors for long-term surveillance, for instance. The shift means that we need to focus on maximizing resource efficiency rather than just maximizing performance. They emphasize that in order to achieve efficient use of energy and communications, systems should be distributed, co-operative, and opportunistic (e.g. power-down mechanisms, or systems with multiple states [1]).

Sensor networks are inherently distributed systems. They have many areas of application, including tracking by multiple sensors [4], and structural health monitoring [9], to mention a few. In these applications, there is a concern whether the long-term operation of such systems is feasible at all, considering that there is a large cost for maintenance, to change the batteries, for instance. In the area of structural health monitoring, there are solutions to seek for for parsimonious models in order to minimize the amount of communication [10] between the sensor nodes.

Mobile sensing has become commonplace in everyday smart phone applications, however, the sensing consumes a fair amount of energy [16]. Since energy consumption may become an issue, researchers [3] created a simulation library to simulate a real-world deployment scenario in order to predict resource use in smart phones and wireless sensor networks.

A more theoretical perspective on computation and the energy needed to to compute (if at all) is discussed in [13], where the trade-off between time to perform a computation and energy needed to compute is discussed.

One line of work towards more resource-aware computation is to use lower precision of floating point numbers than the standard [8] used in digital computers. One such work makes use of limited-precision floating point numbers in the context of naive Bayes classification [11]. The research questions are then to quantify the overall compromise in the accuracy of the algorithm in the classification setting.

Overall, most of the research on energy efficient algorithms consider optimizations and savings that are algorithm specific. It means that optimizing every next algorithm starts from the beginning.

4. ANALYSIS OF ALGORITHMS AND ELEMENTARY OPERATIONS

From the data analysis algorithm design perspective energy efficiency may be improved in three main ways:

1. approximate computation,
2. revising the necessity of operations within an algorithm,
3. more efficient implementation of operations.

The first two approaches would need to be tailored for concrete algorithms. For example, if we want to optimize k-means clustering algorithm, we would need to decide, which computations within this algorithm can be done approximately, and which operations may not be necessary. Moreover, optimizing single algorithms is tedious, and the gains in energy efficiency may vary from algorithm to algorithm. For some algorithms it may be difficult or impossible to come up with further computational optimization. Hence, we find the third approach to optimization to be promising.

Instead of optimizing individual algorithms, we can try to optimize elementary operations that are commonly used by many algorithms. These operations could be optimized at the algorithmic level and the corresponding processing elements at the hardware level. For example, k-means requires computing distances between two vectors many times. Optimizing the distance computation operation would contribute to a wide range of data analysis algorithms, not only k-means.

The first step towards such an approach would be to identify the most common elementary operations used in data analysis. For a rough approximation of such ranking we conduct the following survey.

We analyze the top 10 data mining algorithms [14], recently identified by a panel of research leaders: C4.5, k-Means, SVM, Apriori, EM, PageRank, AdaBoost, kNN, Naive Bayes, and CART. In addition, we consider three more algorithms, commonly used in practical data mining tasks: the logistic regression (classification), the linear regression, and the Fast Fourier Transform (FFT) commonly used in signal processing.

Table 1 lists elementary operations on the left hand side, and each column corresponds to one of the 13 data mining algorithms. A bullet sign indicates the presence of an elementary operation in that algorithm. The bottom line indicates to which data mining task an algorithm belongs: CLA - classification, REG - regression, C&R - classification and regression, CLU - clustering, ASS - association rule discovery, DSP - digital signal processing. The right column gives a weighted score, that we will describe in more detail.

Figure 1: Estimated shares of data mining/machine learning tasks on autonomous devices.

We could simply sum the appearances of each elementary operation and prioritize the operations, that get the largest number of points, assuming that these are the most frequent operations. However, this does not take into account that each of the 13 algorithms is not equally to be used, some algorithms may be used more often than the others. To take the frequency of usage into account we first consider what is the share of each data mining task in practical sensor data analysis using autonomous devices. This allocation is completely subjective, and is based on our personal experiences; however, it provides a baseline for a start. An alternative way to estimate the relative prevalences of different algorithms would be to use an experiment database, see [12].

If we think of a data mining/machine learning processor, optimized for energy efficiency when analyzing data, we allocate shares of usage to different activities as given in Figure 1. Each of the 13 algorithms belongs to one of the categories. We assume that within each category any of the considered algorithms is equally likely to be used. For example, all clustering is assumed to take 35% of the processor time, we have two clustering algorithms (k-means and EM), hence, each of them is allocated 17.5%, which is half of the total allocation. The weighted scores are obtained by the following equation: $S_{weighted} = N \sum_{i=1}^{N} w_i r_i$, where $N = 13$ is the number of algorithms, w_i is the weight of each algorithm, $r_i = 1$ indicates the presence or $r_i = 0$ indicates the absence of a given elementary operation in algorithm i. The weighted scores in Table 1 indicate the frequency of each operation in the presumed general purpose autonomous data mining device, the higher the score - the more important it is to optimize this operation for making it energy efficient. This is a simplified approach to analysis, since we assume that each algorithm requires the same number of operations.

Table 1: Analysis of elementary computations.

Operation	C4.5	SVM	Adaboost	k-nn	Naive Bayes	CART	logistic regression	Lin.reg.	PageRank	k-means	EM	Apriori	FFT	Weighted score
Sum of the vector elements		•	•	•	•		•	•	•	•	•	•	•	11.5
Average of the vector elements			•	•	•		•	•	•	•	•		•	10.2
Sub-setting of values, windowing, find	•			•	•	•			•	•	•	•	•	10.1
Normalization to unity		•		•	•				•	•	•	•	•	9.3
Euclidean distance			•	•			•	•		•				5.2
Logarithm	•		•		•	•					•			5.2
Histogram					•				•		•	•		4.3
Inner product		•	•				•	•						2.9
Square root	•		•		•	•								2.9
Discretization	•				•	•						•		2.8
Entropy calculation, conditional entropy	•		•			•								2.2
Gini index	•		•			•								2.2
Cosine, Sine (due to exp())		•											•	2.0
Sigmoid function (non-linearity due to exp())		•						•						1.5
SVD, Matrix inversion		•	•											1.5
Absolute value														0
Weight (w)	5.6%	5.6%	5.6%	5.6%	5.6%	5.6%	5.6%	5.6%	5.0%	17.5%	17.5%	5.0%	10.0%	
Task	CLA	C&R	C&R	C&R	REG	C&R	CLA	REG	RAN	CLU	CLU	ASS	DSP	

We can see from the table that vector operations take the priority. The first four operations receive by a large margin higher weighted score than the rest. These should have priority for optimization on the hardware level with application specific components and memory addressing.

5. CONCLUSION

Our position is that in order to be able to cope with the new challenges of the growing digital universe, research needs to take a combined view towards data analysis algorithm design and hardware design. A way to do it is to iteratively analyze the algorithms at an elementary operation level, optimize the performance of the relevant elementary operations at the hardware level. As the first step in this direction, we presented a preliminary survey of such operations.

When designing algorithms, or choosing existing algorithms for implementation, data analysis should be aware of energy costs of elementary operations at different hardware designs, and much energy which operation consumes. That would allow to prioritize elementary operations in the hardware design.

This integrated approach to algorithm design opens an interesting and important research directions. The focus should be on understanding the big picture of the available data analysis algorithms, which will allow to develop low-energy solutions for the big data problems in a systematic way.

6. ACKNOWLEDGEMENTS

This research is supported by Academy of Finland grant 118653 (ALGODAN), and ARTEMIS joint undertaking under grant agreement no 641439 (ALMARVI).

7. REFERENCES

[1] S. Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, May 2010.

[2] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Minimizing communication in numerical linear algebra. *SIAM J. Matrix Analysis Applications*, 32(3):866–901, 2011.

[3] M. Buschhoff, J. Streicher, B. Dusza, C. Wietfeld, and O. Spinczyk. MobiSIM: A simulation library for resource prediction of smartphones and wireless sensor networks. In *Proceedings of the 46th Annual Simulation Symposium*, ANSS, pages 8:1–8:8, 2013.

[4] V. S. Cheng and E. H.-C. Lu. Energy-efficient real-time object tracking in multi-level sensor networks by mining and predicting movement patterns. *The Journal of Systems and Software*, 82:697–706, 2009.

[5] IDC. Emc digital universe, 2014. http://www.emc.com/leadership/digital-universe/2014iview/index.htm.

[6] G. Moore. Progress in digital electronics. *Technical Digest of the Int'l Electron Devices Meeting*, page 13, 1975.

[7] G. E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, pages 114–117, 1965.

[8] M. L. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*. SIAM Publishing, 2001.

[9] J. Toivola and J. Hollmén. Feature extraction and selection from vibration measurements for structural health monitoring. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis*, IDA, pages 213–224, 2009.

[10] J. Toivola and J. Hollmén. Collaborative filtering for coordinated monitoring in sensor networks. In *Proceedings of the 2011 11th IEEE International Conference on Data Mining Workshops*, ICDMW, pages 987–994, 2011.

[11] S. Tschiatschek, P. Reinprecht, M. Mücke, and F. Pernkopf. Bayesian network classifiers with reduced precision parameters. In *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases*, ECMLPKDD, pages 74–89, 2012.

[12] J. Vanschoren, H. Blockeel, B. Pfahringer, and G. Holmes. Experiment databases. *Machine Learning*, 87(2):127–158, May 2012.

[13] P. Vitanyi. Time, space and energy in reversible computing. In *2005 ACM International Conference on Computing Frontiers*, pages 435–444, May 2005.

[14] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, Dec. 2007.

[15] Q. Zhao, R. Manohar, R. Ulman, and V. V. Veeravalli. Resource-constrained signal processing, communications, and networking. *IEEE Signal Processing Magazine*, 24(3):12–14, May 2007.

[16] I. Zliobaite and J. Hollmén. Mobile sensing data for urban mobility analysis: A case study in preprocessing. In *Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference*, pages 309–314, 2014.

Unleashing the Power of Information Graphs

Matteo Lissandrini
University of Trento
ml@disi.unitn.eu

Davide Mottin
University of Trento
mottin@disi.unitn.eu

Themis Palpanas
Paris Descartes University
themis@mi.parisdescartes.fr

Dimitra Papadimitriou
University of Trento
papadimitriou@disi.unitn.eu

Yannis Velegrakis
University of Trento
velgias@disi.unitn.eu

ABSTRACT

Information graphs are generic graphs that model different types of information through nodes and edges. Knowledge graphs are the most common type of information graphs in which nodes represent entities and edges represent relationships among them. In this paper, we argue that exploitation of information graphs can lead into novel query answering capabilities that go beyond the existing capabilities of keyword search, and focus on one of them, namely, *exemplar queries*. Exemplar queries is a recently introduced paradigm that treats a user query as an example from the desired result set. In this paper, we describe the foundations of exemplar queries and the significant role of information graphs, and we present several applications and relevant research directions.

1. INTRODUCTION

The typical way of searching documents, objects or structures is through queries [1, 4]. Structured queries describe very accurately the objects of interest but they are hard to formulate. For this reason, several different search models have been studied in the past, including keyword search, similarity search, related search, personalized results, query refinement and query relaxation. However, these types of queries are vague and auxiliary information, such as knowledge bases, query logs or user profiles, is needed in order to improve the quality of the retrieved results [25]. One type of such auxiliary information is information graphs. Information graphs are graph structures like social graphs, knowledge bases, gene networks, etc. A common form of information graph is a *knowledge graph* in which nodes are entities, such as *Google* and *YouTube*, and edges are relationships between these entities, e.g., *Google acquired YouTube*.

A recently introduced query paradigm is *exemplar queries* [19]. Exemplar queries are particularly useful to cases in which the user knows one single element among those that are expected to be in the desired result set, and the system needs to infer the rest of the elements from it. For instance, a traditional keyword query on the World War II will traditionally return documents related to this war. Evaluating the query as an exemplar query will return many other big wars in history, such as the Vietnam War or the American Civil War. In other words, the user "query" is just an example of the elements and inter-relationships of interest that are expected to be returned by the search engine. Exemplar queries are particularly suitable for the case of an investigator, a lawyer, a reporter, a student, or a citizen that needs to perform a study on a topic to which she may not be familiar with, yet has as a starting point an element from the desired result set. Initial studies show that more than 80% of the users would benefit from a service implementing this paradigm [19]. As we discuss below, the exemplar query paradigm is flexible enough to be applied to many different data sources as long as there exists an effective method to convert them into information graphs.

Exemplar query answering is a multi-step task that evaluates an exemplar query and returns the set of results that are considered similar to those the user provided. The first challenge is that a user may not be able, or willing to provide all the details of the example, thus we need first to identify a subgraph of the information graph that better matches the user input. To do so one can exploit information graphs by identifying subgraphs that satisfy the user provided conditions. These subgraphs are referred to as *user samples*. Once the user samples have been retrieved, similar structures need to be found by employing graph similarity techniques. In order to provide different aspects of the results found in the previous step, the structures are refined by adding nodes and edges that belong to the information graph and are connected to those results. Finally, the results are ranked to present first those that more likely match the user's interests. If

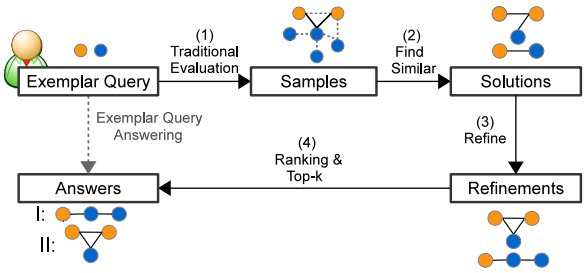

Figure 1: Exemplar query answering.

the number of these results is large then only the top-k can be produced and shown. These steps are graphically depicted in Figure 1 and have been implemented in the *XQ* system [20].

In the following, we provide a detailed description of how information graphs are used in these steps, and present a number of other applications scenarios in which informations graphs can be used.

2. EXEMPLAR QUERY ANSWERING

We define an *information graph* as a graph with labels on the edges and names as identifiers on the nodes. More specifically, an information graph is a pair $G : \langle N, E \rangle$, where N is the set of nodes and E is the set of edges. Each edge has a label l from the set $\mathcal{L}$ of infinite label names.

In *exemplar queries*, the information provided by the user is considered as a member of the desired answer set. Evaluating an exemplar query means discovering what are the main characteristics of that example and finding other similar structures in the information graph. Given a function $eval(Q_e)$ that evaluates the query Q_e on the information graph G and returns a set of samples, we define the evaluation of an exemplar query as follows.

Definition 1. *The evaluation of an* exemplar query Q_e *on an information graph* $G : \langle N, E \rangle$*, is the set* $\{a \mid \exists s \in eval(Q_e) \wedge a \approx s\}$*, where* a *and* s *are structures in* G *and the symbol* $\approx$ *indicates a similarity function.*

2.1 Finding Samples

A natural way of searching is keyword search. However, keyword queries are ambiguous in nature and as a result, many methods have been proposed to evaluate keyword queries into subgraphs of an information graph. Examples of such methods are semantic expansions [6, 23] and keyword to graph query translation [13, 25].

Evaluating a keyword query into a graph query requires several steps. The first is to split the query into chunks of one or more words. External information, such as document corpora, are usually exploited to mine the frequency of co-occurring words. The chunks are then assigned to candidate nodes in the graph by exploiting simple matching techniques on the node labels alongside some thesaurus. The next step connects the candidate nodes to form subgraphs. Since several subgraphs are found, probabilistic ranking (e.g., using random walks) is performed to retrieve the most likely *user samples*.

Relationships between concepts are often explicitly expressed in the user query. Some existing approaches explicitly elaborate on this fact either through similarities [2], or by employing mathematical models [3]. We explicitly take them into account to improve the quality of the sample retrieved from the information graph [14].

2.2 Finding Similar Structures

Traditional query answering on graphs [15, 27] is not directly applicable to exemplar queries, as it focuses on finding the best subset of nodes matching a given graph-query. Instead, in exemplar queries, structures *similar* to the user sample need to be retrieved in the information graph. We refer to these similar structures as *solutions*. Although any similarity measure can be used, a natural one is based on graph node-and-edge isomorphism [19]. While previous works are node-label preserving, i.e., they match both nodes and labels in the sample with those in the graph, exemplar query similarity preserves only the edge-labels and their structure.

Figure 2 illustrates a knowledge base and the sample: "*Google founded in Menlo Park acquired YouTube*" denoted as S in the figure. Exploiting the graph isomorphism similarity with the sample S, one can retrieve as solutions both $A1$:"*Yahoo! founded in S. Clara acquired del.icio.us*", and $A2$: "*GM founded in Flint acquired Opel*".

In the following, we briefly describe three algorithms for the identification of such structures. The first is an exhaustive algorithm that seeks in the graph the structures isomorphic to the user sample. The second is a fast pruning technique to restrict in advance the search space. Finally, the third algorithm is an approximate solution based on the Personalized PageRank measure. These techniques enable real-time exemplar query answering (i.e., responses in less than 1s), even when using the entire Freebase knowledge graph [19]).

Exhaustive solution. We assume as input a connected graph $S : \langle N_S, E_S \rangle$ that represents a sample. The notion of isomorphism is exploited to implement the similarity function $\approx$ in Definition 1, i.e., to find every subgraph $G' : \langle N_{G'}, E_{G'} \rangle$ of G that is

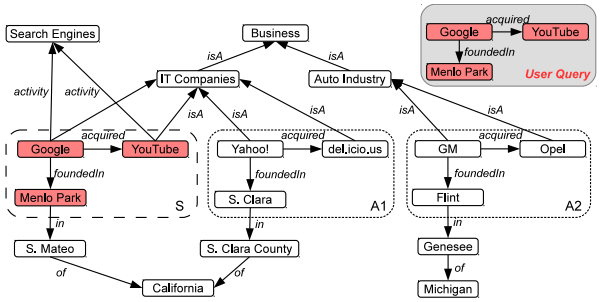

Figure 2: Answers to an exemplar query on a knowledge base

isomorphic to S. The set of all such subgraphs are the solutions. To do this, the sample S is compared with every other subgraph in the database. If an isomorphic graph is found, then it is added to the set of *solutions*. The algorithm initially starts from a node of the sample and finds a matching node in the information graph. It iteratively visits the nodes in the sample and in the graph until either all the nodes have been visited and a subgraph matching the sample has been found, or it is not possible to retrieve an isomorphic mapping. This algorithm is exponential in the number of query nodes.

Fast Exact Solution. Since the exhaustive solution is not efficient for large graphs, a better method for exploring the solution space is needed. The proposed method employs an efficient technique for comparing nodes, and an algorithm for effectively rejecting node pairs that do not participate in any isomorphic mapping. The idea is to store a compact representation of the neighborhood of each node. This representation is called d-neighborhood and referred to as $\mathbb{N}_d(n)$, i.e., nodes and edges that are at a fixed distance d from each node.

For every node in the database we compute a table consisting of the number of nodes that are reachable from that node at some specific distance and with a path ending with an edge labeled ℓ. In other words, for a node n, for every label ℓ and for every distance i we keep the cardinality of the set $W_{n,\ell,i} = \{n_1 | n_1 \xrightarrow{\ell} n_2 \vee n_1 \xleftarrow{\ell} n_2, n_2 \in \mathbb{N}_{i-1}(n)\}$.

This compact representation is then exploited in the algorithm through the concept of *simulation*. A graph simulates another graph if there exists a way to map each transition on the first graph with a transition in the second. Consequently, if a graph cannot simulate another graph, then they cannot be isomorphic. The main idea of our approach is to perform multiple simulations with the user sample on the database graph, comparing only the d-neighborhood representation of the nodes, without having to actually visit their neighborhood. The result is a method that operates much faster than the exhaustive solution, while providing the same, exact answers. While the algorithm is still exponential at worse, it is practically much faster than the exhaustive solution. The d-neighborhood can be computed and stored in advance in $\mathcal{O}(d|N||L|)$ space, where $L \subset \mathcal{L}$ is the finite set of edge-labels in G. Provided that the d-neighbor is sparse, efficient indexing techniques can be employed [15].

Approximate Solution. In order to further restrict the search space to a meaningful set of nodes that are likely to contain the top-k solutions for the user, a principled approach based on the Personalized PageRank (PPV) algorithm is considered. Since user preferences are expressed through the exemplar query, the PPV has to be computed over nodes in the information graph, instead of webpages. Instead of treating each edge equally, as in the original PageRank, and in order to better capture the semantics of the edge labels we propose the Adaptive Personalized PageRank Vector (APPV) method. APPV $\boldsymbol{v}$ is defined as the stationary distribution of the Markov chain with state transition given by $\boldsymbol{v} = (1-c)\boldsymbol{A}\boldsymbol{v} + c\boldsymbol{p}$, where $\boldsymbol{A}$ is a *non uniformly distributed* adjacency matrix and $\boldsymbol{p}$ is a preference vector computed from the user query. To compute this value fast, we apply an approach similar to the *weighted particle filtering procedure* [16], extended to take into account non-uniform edge weights. Finally, the algorithm returns the subset of G containing only those nodes with APPV score higher than a threshold. Isomorphic structures are then found in this subset.

2.3 Graph Query Refinement

If the user provides a general exemplar query with an exploratory intent, then query refinement is needed. Moreover, since a vague query usually retrieves a large number of *solutions*, query refinement can be used to group them offering a small set of more specific aspects that capture the structures contained in most of the solutions. These two common situations are the main motivations for the study of query refinements in information graphs. Query refinement has been studied in the web [7] and relational databases [21], but no straightforward adaptation exists with graph data sources.

A refined query is a more restrictive query than the original one, which retrieves a subset of the results of the original query. Therefore, finding refinements to a graph query means retrieving a set of supergraphs of the original query. Given a query Q we say that Q' is a reformulated query if (i) it is a connected graph, and (ii) there exists a subgraph

isomorphism from Q to Q'. Given a set of graphs $\mathcal{D} = \{G_1, ..., G_n\}$[1] and a query $Q : \langle N_Q, E_Q \rangle$ the results of the evaluation of Q over the set $\mathcal{D}$, is the set $\mathcal{R}_Q$ of subgraphs isomorphic to Q.

Given that the number of reformulations is exponentially large, only a meaningful subset should be returned. In order to retrieve relevant reformulations, the objective function sums two different contributions: the number of $\mathcal{R}_Q$ captured by the candidate reformulations (*coverage*), and the number of diverse results in each result set of the reformulated queries (*diversification*).

The coverage is a function *cov* that takes as input a set of reformulated queries R, and computes the number of results captured by R, that is, $cov(R) = |\bigcup_{Q \in R} \mathcal{R}_Q|$. The diversification is represented by a function *div* that takes into account the number of elements that are in the result set of Q_1 and not in Q_2, namely $div(Q_1, Q_2) = |\mathcal{R}_{Q_1} \cup \mathcal{R}_{Q_2}| - |\mathcal{R}_{Q_1} \cap \mathcal{R}_{Q_2}|$. The final score of a set of reformulated queries is the linear combination of coverage and diversity

$$f(R) = cov(R) + \lambda \sum_{Q_1, Q_2 \in R} div(Q_1, Q_2)$$

where $\lambda \in [0, 1]$ is a parameter that regulates the diversification of the result set R. The set R that maximizes the two factors of $f(R)$ is selected.

Fast Exact Solution. The optimization problem can be reduced to an instance of Max-Sum Diversification, known to be **NP**-hard. For this problem, there exists a $\frac{1}{2}$-approximation greedy algorithm [8] that runs linearly with respect to the input size. The greedy algorithm selects at each step the query that maximizes the *marginal potential gain*, i.e., the difference between the value of the function f, adding one extra element and the current value of f. Although the greedy algorithm works well in the normal set-covering formulation, in which the sets of elements are known in advance, the generation of the reformulations requires an exponential number of operations, as we consider all the possible edges that can be added to the current query Q.

Therefore, an algorithm that allows fast and effective pruning of the search space is necessary. The algorithm should remove the query refinements that definitely do not participate in the optimal solution. Since any reformulated query generates other reformulations just by adding one extra edge, the algorithm should find, for each reformulated query already computed, an upper bound to the marginal potential gain. This algorithm will then preserve the correctness of the approximation, yet allowing for an effective pruning of the unpromising reformulations. A further optimization can be introduced expanding at each step the reformulated query with the maximum upper bound.

[1]A set of graphs is obtained considering a neighborhood of the solutions found in the previous step.

2.4 Ranking Exemplar Results

In order to rank the exemplar query results, the structural similarity as well as the closeness of the solution from the sample are taken into account. For the former, a metric that is based on a vectorial representation of nodes using its neighborhood is used [15], and extended to capture the differences among nodes that emerge when taking into account the edge-labels of the neighbors, denoted with $\mathcal{S}$. For the latter, the APPV values are used, as they provide information about the distance of any node from the nodes of the sample, and consequently the distance of each solution from the sample. The final score is the linear combination

$$\rho(n_s, n) = \lambda \mathcal{S}(n_s, n) + (1 - \lambda)\boldsymbol{v}[n], \qquad (1)$$

where n_s is a node in the sample, n a node in the result, $\boldsymbol{v}[n]$ is the APPV score and λ is a user-defined parameter that regulates the amount of diversification in the results. For instance, in Figure 2 the ranking function will rank $A1$ higher, because it is closer to the sample S than $A2$.

3. APPLICATIONS

Exemplar queries find various applications in several different contexts. Below we discuss four such applications, namely, related queries, document search, seed-based search, and recommender systems.

3.1 Related Queries

Keyword queries are typically vague in their semantics, and a lot of work has been devoted in proposing alternative queries that generate results, which are more likely to match the user interests. *Query refinement* [18] and *query relaxation* [21] are two techniques that make a user query more specific, or more generic, respectively, in order to better match the user expectations. A similar concept is the concept of *related queries* [1]. The idea is to offer to the user a number of alternative queries that retrieve concepts that are related to those that the user asked about, or concepts that the user may be interested in finding more information about. Identifying related queries is a challenging task. Various methods, such as the exploitation of user query logs [1], large document corpuses [4] and knowledge bases [23] have been proposed. Exemplar queries can be seen as a complementary methodology that

suggests alternative queries. Having identified the user samples in a knowledge graph (e.g., Freebase) from the keyword query that the user provided, finding the related structures is like finding related concepts, from which queries can be formulated and then presented to the user as related queries [19].

3.2 Document Search

Document search is the core task of web search engines, as well as that of many document databases. Given a keyword query, the search engine retrieves the documents that are related to these keywords, which typically means that the documents have these keywords in their content. Current solutions are based on a plethora of different technologies, like topic-models [5] and click-models [10], and exploit various data sources, such as query-logs and knowledge bases. Exemplar queries can offer new ways to enrich this type of query results. Evaluating the keyword query through a traditional evaluation process, and at the same time through the exemplar query evaluation process described previously, will lead to a much richer answer set. This answer set will contain documents that may not include the original query keywords, but will involve concepts that are similar to those expressed in the original query. Entity linking [11, 22] can translate a document into a graph sample to be used in a knowledge graph as an exemplar query. This type of search has often been coined as semantic search, since the user is looking for semantically related documents, even if their direct structural relationship is not explicit. Furthermore, what is critical in this type of search is to be able to identify user intentions through the actions (e.g., previous searches) that a user has performed. This can be achieved by exploiting a graph representing previous knowledge on sequences of actions leading to the achievement of some goal [24].

3.3 Seed-based Search

Documents in document collections may be associated (related) in ways not always expressed through syntactic and semantic content similarity. Looking for this kind of similarity is encountered in many practical scenarios. For instance, in the case of technical blogs, in which users describe solutions to various problems that have faced, a user may be interested in finding a solution to a technical problem and uses a keyword search to identify responses from users that have faced different (but of similar nature) problems. Keyword based similarity will not work in most of such cases. Exemplar queries can help by considering a specific solution as an example and then searching of similar cases. To find such similarities, entity identification [11] can be used to turn the unstructured text document into a structured graph, and this structured query graph can then be matched against a knowledge graph.

Instead of seeing a document as a set of interconnected entities, one can see it as a set of segments, each one used to serve a specific communication message. A segment may be divided further into sub-segments that serve more specific purposes. At the end, the document, fragmented into segments and subsegments has been turned into a tree where the nodes represent the segments. Given a sample document, finding similar documents may turn to be a task of finding documents with the same segment structure or type of segment structure, even if the content of the segments are very different.

3.4 Recommender Systems

Recommender systems try to make item suggestions from an item set, or rank the results of a user query, based on the personal preferences of the user (according to her previous selections in the interaction with the system). This kind of decisions are based on some form of similarity to the properties of the items in the previous selections, as this is computed using a number of different metrics [26]. One can formulate the set of all the different properties of the items and organize them into some graph structure that represents their one-to-one relationships either based on their nature, or on the degree of preference by the various users. Then given some selection made by the user, additional suggestions of similar items can be made, through an exemplar query evaluation method that uses the item features graph as an information graph.

4. EXTENSIONS

The exemplar query answering framework presented previously can be extended in several directions, in order to offer more advanced services.

Multiple exemplar queries. The user may provide multiple examples of the same desired result set and this information can be used to more accurately identify the user needs. Although multiple query processing has been studied in relational databases [9], the exemplar queries context is completely different: in this earlier work, the results for a single, or multiple queries do not change, while in exemplar queries the results become more specific as the number of example queries increases. Moreover, the straightforward solution that computes the intersection between the result sets is not directly applicable when there is no common edge labels in the input samples. A recent study computes the rele-

vant neighborhood of the samples in order to find intersections among them [12]. However, this does not provide a clear semantics for multiple exemplar queries, thus the problem needs to be investigated further.

Approximate exemplar queries. Assume that a user looking for companies in California provides the query *"Google, S. Mateo"*. The identified sample connects *Google* to *S. Mateo* using the edges *foundedIn, in* as in Figure 2. The query, through graph isomorphism, retrieves companies founded in the bay area, that does not perfectly match the user intent. Approximate query answering could solve the problem, retrieving companies not only founded in California but also based in California, employing some similarity between edges. Therefore, extending the similarity measure to handle looser matchings is important. Several adaptations of the current task can be employed, such as approximate queries [17] and bi-simulation instead of graph isomorphism. Due to the large number of results, graph indexing techniques have to be adopted in order for the system to provide timely answers.

Personalization and dynamic ranking. The ranking function presented earlier does not take into account user preferences. User preferences can be mined from query logs, or other sources, and offer the basis for personalized results ranking. The Personalized PageRank (Section 2.2) can be extended to include any probability in edges and nodes, thus reflecting the preferences of the user. Moreover, without employing any approximation, the computation of the node similarity based on the neighborhood can take into account various preferences.

5. CONCLUSIONS

Information graphs are valuable and important source of information, as they embed knowledge in a simple graph structure. The use of information graphs to answer exemplar queries opens interesting research directions that cannot be addressed by existing technologies. This synergy between exemplar queries and information graphs forms the basis of the presented exemplar query answering framework. We argue that this framework is powerful and versatile, able to improve existing solutions by enriching the expressiveness and the ease of use of many search systems, and to increase user satisfaction. Finally, we discuss how this framework can be applied to several different scenarios, such as related queries, document and seed-based search that constitute the basis of modern search engines.

References

[1] A. Anagnostopoulos, L. Becchetti, C. Castillo, and A. Gionis. An optimization framework for query recommendation. In *WSDM*, 2010.

[2] S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado, and Y. Velegrakis. Keyword search over relational databases: a metadata approach. In *SIGMOD*, 2011.

[3] S. Bergamaschi, F. Guerra, S. Rota, and Y. Velegrakis. A hidden markov model approach to keyword-based search over relational databases. In *ER*, 2011.

[4] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. In *SIGIR*, 2011.

[5] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3, 2003.

[6] R. C. Bodner and F. Song. Knowledge-based approaches to query expansion in information retrieval. In *Canadian AI*, 1996.

[7] P. Boldi, F. Bonchi, C. Castillo, and S. Vigna. Query reformulation mining: models, patterns, and applications. *IR*, 14(3), 2011.

[8] A. Borodin, H. C. Lee, and Y. Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *PODS*, 2012.

[9] U. S. Chakravarthy and J. Minker. Multiple query processing in deductive databases using query graphs. In *VLDB*, 1986.

[10] F. Guo, C. Liu, A. Kannan, T. Minka, M. J. Taylor, Y. M. Wang, and C. Faloutsos. Click chain model in web search. In *WWW*, 2009.

[11] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: a graph-based method. SIGIR, 2011.

[12] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri. Querying knowledge graphs by example entity tuples. *CoRR*, abs/1311.2100, 2013.

[13] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *ICDE*, 2008.

[14] Z. Kefato, M. Lissandrini, D. Mottin, and T. Palpanas. Keyword query to graph query. *DISI Technical Report*, 2013.

[15] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao. Neighborhood based fast graph search in large networks. In *SIGMOD*, 2011.

[16] N. Lao and W. W. Cohen. Fast query execution for retrieval models based on path-constrained random walks. In *KDD*, 2010.

[17] W. Lin, X. Xiao, J. Cheng, and S. S. Bhowmick. Efficient algorithms for generalized subgraph query processing. In *CIKM*, 2012.

[18] C. Mishra and N. Koudas. Interactive query refinement. In *EDBT*, 2009.

[19] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar queries: Give me an example of what you need. *PVLDB*, 7(5), 2014.

[20] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Searching with xq: the exemplar query search engine. In *SIGMOD*, 2014.

[21] D. Mottin, A. Marascu, S. B. Roy, G. Das, T. Palpanas, and Y. Velegrakis. A probabilistic optimization framework for the empty-answer problem. *PVLDB*, 6(14), 2013.

[22] D. Mottin, T. Palpanas, and Y. Velegrakis. Entity ranking using click-log information. *IDA J.*, 17(5):837–856, 2013.

[23] V. M. Ngo and T. H. Cao. Ontology-based query expansion with latently related named entities for semantic text search. In *IJIIDS*, volume 283. 2010.

[24] D. Papadimitriou, G. Koutrika, Y. Velegrakis, and J. Mylopoulos. Goals in Social Media, Information Retrieval and Intelligent Agents. In *Proceedings of ICDE*, 2015.

[25] J. Pound, A. K. Hudek, I. F. Ilyas, and G. Weddell. Interpreting keyword queries over web knowledge bases. In *CIKM*, 2012.

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[27] X. Yan, P. S. Yu, and J. Han. Graph indexing: a frequent structure-based approach. In *SIGMOD*, 2004.

Kian-Lee Tan Speaks Out on How to Build a Strong DB Group without Pushing Students Hard

Marianne Winslett and Vanessa Braganholo

Kian-Lee Tan
http://www.comp.nus.edu.sg/~tankl/

Welcome to ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today[1] *we are at the Advanced Digital Sciences Center in Singapore. I have here with me Kian-Lee Tan, who's a Provost's Chair Professor of Computer Science at the National University of Singapore, where he's also the Vice Dean for Research in the School of Computing. Kian-Lee received the President's Science Award in 2011. He has served as co-editor in chief of the VLDB Journal and editor for IEEE Transactions on Knowledge and Data Engineering, and a PC chair for VLDB and ICDE. His PhD is from the National University of Singapore. So Kian-Lee, welcome!*

[1] This interview was conducted in 2011.

So why did you receive the President's Science Award?

Actually, the PSA is an annual award. It is Singapore's most prestigious award given to researchers in recognition of their outstanding contributions in basic research. It is actually a team award, rather than my award. For 2011, Beng Chin and I have the honor of receiving this award for our research on Peer-to-Peer Based Data Management. And, of course, thanks to you for nominating us!

You're welcome! So, that's a great lead into my next question. Which is, so far, peer-to-peer databases haven't had a big impact in industry. Do you see a killer app for them?

Actually, no. To be frank, we were quite disappointed that we were not successful in bringing this technology into industry. We have tried talking to some people, particularly the clinics in Singapore, but I guess they are more concerned with security. So we were not very successful in bringing this technology out. But nevertheless, Beng Chin actually founded a company called BestPeer, that is making use of this technology and he seems to be making some headway in China. So, exactly what is being done is being kept tight-lipped, since it has to do with some government agencies in China.

So not health care?

Right, not health care.

I see…

So there's really no "killer app", but there will be organizations that are interested to look at it.

Mike Stonebraker says that MapReduce is a major step backwards. Do you agree with him?

I don't think so. I mean, you have to look at the objective of MapReduce, right? It's not intended for database applications. The fact is that many people are using MapReduce, especially in the industry, so it is quite a successful framework. From that perspective, I suppose he's referring to database management systems. He's comparing to database management systems. So in that case, I don't think it's right.

Okay, wrong comparison. So, nowadays companies like Microsoft and Google have lots of money, they have hundreds of researchers with PhDs and they have all the data that anybody could want. So how can we as academic researchers compete with those places?

Don't throw everything into a single basket.

First, I think there are lots of problems to be solved, beyond what Google and Microsoft are doing. Of course, they have the benefit of real life data. For us, perhaps we need to wake up and work on some real applications. For example, in NUS now, we have a number of big grants that allow us to build test beds. For example, one of them is on megacities. The government has agreed to set aside some area for us to build our sensors both in China as well as Singapore. So, it's a joint project between China and Singapore. With that, we'll be collecting real data and hopefully we'll be able to do something that is both relevant and impactful.

What kind of data will you collect?

This particular project has to do with air pollution and water pollution. In some sense you can look at it as streaming data. They'll be sensing the pollutants in the air, in the water, and then this will be transmitted to the central live server for real-time processing.

That's an interesting application. So what are the research challenges that you see so far in that?

I think the main thing for us is, we need to understand the domain. And then after that, we need to ensure that this information is collected in real-time. Results need to be presented in real time. So in our part…the contribution, rather, the part that SoC plays is to build a backend server that will be able to process this streaming data quickly and provide information to those who need them in real time.

Okay. Great! So you've been in NUS ever since you were a teenager: Undergrad, Masters, PhD and then on to being a professor! So at that point in time, NUS wasn't known for its research, when you got your PhD. So many of your peers probably went overseas for their PhDs. So how did staying at NUS affect you? In your career?

I don't think it affected much, in terms of recognition. I guess the reason is because we started off well. I mean, my advisor began by publishing in either ICDE or VLDB. So, there's already a certain bar that he has set. And when Beng Chin joined, he also encouraged

us to submit to all these prestigious conferences and journals. So from that perspective I think people know our work and know us. Personally, I think, at the end of the day, nobody remembers or nobody cares where you get your PhD from. I probably don't know where you received your PhD. You may not know where (I received my PhD) until now. I think people recognize your work rather than where you received your PhD.

So the database group at NUS is very well known today and you played a key role in building that up. Can you share your insights in how to make that happen?

Actually, I'm not the only one to take on a key role. I think it's a team effort. I mean, there are other colleagues in the DB group. There are a few things that I think we have done. One is, we actually rank conferences. That means, SIGMOD, VLDB, ICDE are considered Tier 1. And the other conferences are Tier 2, Tier 3. In this way, we effectively encourage our people to submit to Tier 1 first. So everybody, including our students now, whenever it comes to the deadline for SIGMOD, ICDE, VLDB, everyone wants to submit something there.

And then, before you ranked them, what were people doing? So how did that change their behavior? What were they picking to send their work too?

I suppose it was whatever deadlines were available. So now, people may hold back.

Interesting! Did you do other things too?

Yes, I think another thing we have done is established collaboration with overseas researchers. We send our students over for internships. Given that these are better students, good students, to some extent, it helps to establish our credibility. I think we try to increase our visibility by being more professionally active.

Do you mean like in the societies and the journals? Or organizing conferences? Or?

Right, as PC members and things like that. In terms of recruitment, nowadays it is not so much on paper publications that we are looking at.

Oh? What do you look at?

Well, I suppose the impact of the work. I think we have published well, and we would like to have impact beyond just paper publications.

So for a young person, how do you judge the impact?

I guess by the quality of their work. We assess what they have done for their PhD and whether it is a strategic direction that we are interested in. In particular, we are very interested in recruiting those who build systems, so we are more systems oriented now.

Now, you don't like to travel. I'm told you don't even go to conferences.

That's wrong! I attend one conference a year.

Oh okay, well is it a conference in Singapore? Or a conference far away?

Not far away. Usually I attend once a year. And for the next six years, I'll be attending VLDB, because I'm the EiC for the VLDB Journal. Recently I just joined the VLDB Endowment. So there's a reason for me to travel. I try to minimize traveling.

Okay, well then how come DBLP says you have 182 co-authors if you're not traveling?

I actually looked through the list. I think most of them are students at NUS.

That many?

Quite a large number. For example, I work with Beng Chin, so Beng Chin's students are in the list. I work with Anthony Tung, and Anthony's students are in the list. Some of them are interns from Chinese universities. So there's quite a lot of interns that come from Chinese Universities. In addition, I've been around for 20 years.

I've been told you catch your bus at 5pm every day. Even if there is a deadline. And that you're also active with your church and with your family (you have 2 kids). So how do you manage to balance all these things?

First, I actually arrive at work at 7am or so, right? Otherwise my boss would be upset with me. Yeah, the reason for leaving early is to beat the traffic. Otherwise my traveling time would be very long. So I always do it to hopefully always get a seat in the bus.

Okay so how long is your traveling time?

At non-peek hours, it is probably 45 min, and at peek hours it could be as long as 1.5 hours.

So you must leave, if you get there at 7am, you must catch the bus at 6:15am…

Yes, that is right.

6:15am?!

Yes.

(laughter from both)

And then, it's still a 45-minute ride. Well coming from Champaign, 45 minutes to me is practically forever. Actually 45 minutes, that's all the way across Singapore, right?

Yeah.

How do stand such a long ride?

Actually I find my journey to be quite productive. Usually I read in the bus. I can review a paper. I read books. So I think it's not wasted time. All I need is a seat in the bus.

Okay, and is it the same when you go home at 5?

Right, that's right.

So they're not too crowded at that point. So really, your workday is 6:15am to 5:45pm and that is a pretty long day.

Yup, I'm glad to hear that.

Wait, unless some people work late at night also.

Yeah, I think the students are expected to work from 10 to 10. But that doesn't mean I stop to work at 5PM.

We need to wake up and work on some real applications.

Then how do you fill the other stuff? The church thing and family time…?

So, I keep my Saturdays free for sure. I try not to be involved in any other activities, except with my family. If possible, no church activities, no office activities. I spend my Sundays at church, from morning until 6PM. So, I have five full-time days related to office work. If there is a need to, I may have to review a paper at home after 5PM. In fact, for the last SIGMOD, I worked until 3AM.

Woah! Okay. And what about your kids? If they're in high school they're working until midnight themselves, aren't they?

No, my son is going for the University next year.

Oh okay!

My daughter is coming to Secondary 2, which is 14 years old.

I've been told that you never push your students hard. But some people think they have to push their students hard. So, how does that work for you?

Is that a compliment? Or….? (laughing)

A complaint? No no! I promise the person who suggested I asked this thought it was great, so they weren't complaining.

I think it's a matter of style. For myself when I was pursuing my PhD, my advisor did not push me. So I was pretty much independent, but I saw my advisor regularly. So in some sense, I expect the same thing of my students. I make it clear to them that they should be independent, that they have only 4 years of scholarship. To some extent, that is indirectly pushing them. So if they don't produce within 4 years, if they can't graduate within 4 years, they might not have financial support beyond that. So far, it works. So I don't like to scold people.

So if you were a scolder does that mean… would your DBLP… would you produce more? Or do you think it's a matter of style and it doesn't directly have any impact?

I think the students already have a lot of pressure, especially from the DB group. I mean, there are quite a number of students who publish in the so-called Tier 1 conferences and journals, and those who have not are really pushing themselves hard.

So you've had a very successful career. What's been your biggest failure?

At one stage, I'm interested in exploring bioinformatics. I actually graduated two PhD students in bioinformatics research. But I view this is still as a failure in the sense that what I've done is still more from the CS point of view, rather than contributing more towards the science. So actually, I have a joint appointment at Genome Institute of Singapore. And

I think people recognize your work rather than where you received your PhD.

after two years, I have not been able to interact well with people there. I guess partly it's my fault, in the sense that I didn't really interact well with them. Because in the initial stage, the scientists there were more interested in getting the CS people to develop tools for them. They are doing manual stuff and they want to automate it. So from my point of view, these are more engineering. So in the end, it did not work out. I wish that more could have been done.

So probably some of our readers are trying to collaborate with biologists, so what advice do you have for them? What's the right way to connect with the biologists? How can you make it work?

I think you must be prepared to spend more time with them, to familiarize with their domain. Perhaps to even be prepared to do some engineering work for them in the process.

I hear that you're a collector. What do you collect?

I collect Coca-Cola cans and bottles from different countries.

How many do you have?

Oh I didn't count, but usually when I go to a new place, if I find a can that I don't have, I'll buy it and bring it back. And now that my students are aware of it, sometimes when they travel, they'll bring one for me too.

I see. Because I was going to say if you don't travel, your collection would be very small! Maybe twelve cans?

No, I have more than that. My students will bring them back for me.

I see… okay. So why Coke cans?

Why Coke cans?

Are you a Coke drinker?

Not really, I drink coffee.

(laughter from both)

So why Coke cans… I supposed it is just to be different from others.

Okay… it is different! Why do you like research?

You get to do what you like. You get to develop new knowledge, create new knowledge that will be hopefully useful to society. We have that flexibility to do whatever we want.

Okay good! Why do you keep your clocks 15 minutes ahead?

So I can leave at 4:45 now (laughs). No, I'm very particular about punctuality. I don't like to be late for meetings. I think it is a mark of respect to others. I don't like others to wait for me, and I also don't like to wait for others. Even with my wife, when we go shopping and if I arrange to meet her at some place after a certain time -- if she's late, I'll be upset too.

But you'll think she's late because your watch is 15 minutes ahead!

No no! I will know how to adjust.

But somehow it helps keep you on time?

Right.

I have been told that the emails you send are short enough to be a tweet. "See me –KL". "How? –KL". "Status? –KL". So why are your emails so short?

Usually I actually write something longer, but in the end I always shorten my emails after writing it. I think in this case, there is a context behind it, which the recipient will know. I suppose this comes from my students, so we must have discussed something before and so I expect a response. So they would know what I'm talking about.

And their answer is very long?

Could be, but usually I ask them to see me rather than looking through their response.

Do you have any words of advice for fledgling or mid-career database researchers or practitioners?

I would say "don't throw everything into a single basket". Don't just work on one single problem. It would be too focused. Sometimes it's good to explore other fields at the same time and other areas at the same time. I mean, this is what I tell my PhD students. They should do more than what needs to be included in

their thesis. I guess it's the same as my own experience. For my PhD thesis, for example, I did more work than what is included in my thesis. I think that will give them a broader view of what is happening in that area, rather than being too narrow.

I think that's very good advice because my PhD advisor only gave me two pieces of advice and that was one of them. Although his was a simpler version, he said, "Always do two things".

Okay, among all your past research, do you have a favorite piece of work?

I think my favorite earlier work would be my PhD work on query optimization in parallel databases. But if you talk about my more recent work, our work on query authentication is something that I like. I think one aspect that has still not received attention is the concept of minimality. What it means is that "only answers are returned". Most of the current work requires exposing certain boundary points. So I think ours is the only work that doesn't require exposing boundary points.

I see, so you came up with a way so that they can verify the answer is complete without showing some things that are not part of the answer. And so, why don't you like the "show the boundary" approach?

So effectively, it is releasing more information than is required for the user.

So it's not exactly Access Control, but yeah, I understand.

If you magically had the time to do one additional thing at work that you're not doing now, what would it be?

I actually wanted to write a book every three years.

Really?!? Wow.

Right from the beginning, I actually co-authored a book every three years. After that, I stopped. So I think if I had time I would be …

What would the topic be if you write it yourself?

I think at this moment, I am interested in security. Database security. I'm teaching a course on security, and there is really no good textbook out there. So it is something I would be interested to consider.

If you could change one thing about yourself as a computer science researcher, what would it be?

To be more pro-active in talking to other researchers that are not in database research. In that way, more interesting research ideas could come out of it. For example, now, we're doing some work on PCM (Phase-change-memory). If I could spend some time just talking to colleagues that are working in that field, maybe our idea could have been more interesting.

So what's the relationship between phase-change memory and databases?

I suppose it's the technology, right? Phase-change-memory allows you to keep more data in memory, while still facilitating the random access.

Thank you very much for talking with me today.

Thank you Marianne.

Sudipto Das Speaks Out on Scalability and Elasticity of Database Systems

Marianne Winslett and Vanessa Braganholo

Sudipto Das
http://research.microsoft.com/en-us/people/sudiptod/

Welcome to ACM SIGMOD Record's Series of Interviews with distinguished members of the database community. I'm Marianne Winslett and today we're in Snowbird, Utah, site of the 2014 SIGMOD and PODS conference. I have here with me Sudipto Das, who is a researcher at Microsoft and the recipient of the 2013 SIGMOD Jim Gray Doctoral Dissertation Award, which is for his dissertation entitled "Scalable and Elastic Transactional Data Stores for Cloud Computing Platforms". Sudipto's PhD is from the University of California Santa Barbara where he worked with Divy Agrawal and Amr El Abbadi.

So Sudipto, welcome!

Thank you very much!

Tell me about your dissertation

So before I get to the actual dissertation work, let me give you some background of where I started. It was about 2008-2009 when I got started in this area, early in my PhD program. There was a lot of buzz about Cloud Computing, NoSQL, and key-value stores. The Google BigTable paper had come out, Dynamo from Amazon had come out, and there was a lot of buzz about key-value stores and database systems being dead, RDBMS are bad, and whatnot. So my advisors, Divy and Amr, suggested to try to analyze why the key-value stores were so successful and why people were saying that RDBMS were bad. We then realized that you can look at it as a spectrum where on one axis you have consistency guarantees or ACID properties, and on another axis you have scalability. Key-value stores were very high on the scalability axis, but very low in terms of guarantees. RDBMS, on the other hand, have provided very good guarantees in terms of transactions, in terms of access methods, and whatnot, which key-value stores lacked.

The question that we wanted to answer as part of our work was: is there a way to bridge this gap between the key-value stores and relational databases? One of the key insights that we derived by analyzing all these key-value stores, looking through it, and brainstorming about it, is that the key fundamental idea was to limit most of the accesses to a single server or single node, to avoid a lot of distributed synchronization. Distributed synchronization was still being used in a lot of these key-value stores, but they were using it very judiciously. So with this abstraction and with this learning, we tried to see what transactional abstractions could be carried forward while limiting accesses to a single server. So you get all the good properties of key-value stores: you can scale out, you get elasticity, you get high availability, but you can still provide transactions at certain granularity.

Essentially, the first half of my dissertation looks into two different ways of designing such systems. One way is through a statically partitioned database, where you define a specific schema pattern. It is a hierarchical schema pattern that is actually explored in a number of other systems as well. We show that if you have such a hierarchical schema pattern, and if your transactions adhere to and only access a given hierarchy, you can provide efficient transactions while scaling out similar to key-value stores. The other abstraction that we were looking at was, what if these partitions weren't statically defined? So think of it as if an application comes in and dynamically specifies that "here is a bunch of data items on which I want transactional access for a certain period of time." Once such a declaration is provided, the system takes on the responsibility of the transactional access on this group of items during a certain span of time, after which the application says, "I don't need it anymore," and the system is free to do whatever it wants. So we came up with an abstraction called they *Key Group* abstraction, where essentially an application can come and specify a group of data items. We do some distributed synchronization to localize accesses to the data items during the lifetime of what we call the Key Group. Transactions execute on the Key Group. We take on the responsibility of doing it efficiently. Once the application says the group can be deleted, we take on the responsibility of propagating the updates back to the original servers that hosted the key-value pairs, from which the application has formed the group, and life moves on beyond that. So it's a way of dynamically defining your partitions on which you want transactions.

> ***"The PhD is your entire life compressed in 5 years. You'll see ups and downs throughout your life. You'll see ups and downs throughout your PhD".***
>
> ***Divy Agrawal***

After doing the transactions part, one part that was still left was elasticity. In the key-value stores, one of the key selling points is elasticity and so is that of cloud. So the next question that we started to answer was that, can we make (classical) databases elastic as well, while executing transactions? One of the key mechanisms that was missing was this concept of live (database) migration. Elasticity essentially means that when the load goes up, you add a new server, your data spreads out to the new server, and your new capacity gets used. And this is all happening while the system is running, causing minimal disruption to the transactions that are executing.

One of the key mechanisms that enable this is what we call a live database migration. A database is on server *A*. At some point in time, a (system) controller decides that it is time to move this database off to do elastic scaling, and it initiates this live migration while transactions are executing with as little disruption as possible. On the fly, this database gets moved from

server *A* to server *B*. The new transactions get routed to server *B* and from the user's perspective it's minimal disruption. And this can happen the other way around -- when the load goes down, the servers shrink, you have consolidation. That is, it helps both ways. So essentially what we developed was two different mechanisms for providing live migration in two different database architectures. One was a shared-storage architecture, where the persistent data is stored in a decoupled replicated storage system, like HDFS in our case. In another case we were doing replication for shared-nothing, where the persistent data was stored in a locally attached disk on the server itself. So during the course of migration, in the first part, we were just migrating the hot cache, so the read and write transactions would see minimal impact at the destination, whereas in the second case, we were actually migrating the persistent data as well, along with the state of some of the transactions. So the first part [of the thesis] talks primarily about scalability and the second part talks primarily about elasticity. And it is the transactions and cloud platforms that ties them together.

Have you seen industrial interest in that?

That's a great question. There is definitely industrial interest in this area. When we were developing these ideas, a number of key industrial developments happened concurrently that kind of resonate on similar ideas. If you look at Microsoft Azure (SQL) database, which used to be called SQL Azure back then, they had a similar notion of a hierarchical schema, limiting transactions to a single node, and then being able to scale out to a cluster. Google's Megastore has taken a similar direction, where there was, again, a hierarchical schema statically defined, and then transactions were operating on that hierarchical schema. So all of this happened concurrently while I was working on my dissertation with my advisors and my colleagues. So I wouldn't say it was an impact to this line of work, but at least it was gratifying to see that indeed these ideas really work out in practice, and things which have similar insights are actually being deployed in production.

In terms of academic impact, there has been a lot of follow-up work. Especially in migration and in transactional key-value stores, people have followed up on our work. There are a number of citations that we have received for our papers. As far as I know, I'm not aware of any system that directly implements my ideas, or the ideas that were developed in the dissertation, but we never know what's under the hood for a lot of these commercial systems.

So true. Is there anything that you know now what you wish you would have known during your PhD studies and job search?

That's a very tricky question. Let me try to rephrase it a little bit. What I would try to answer is what I've learned during the course of my PhD that actually helps me out, even today. So, there was one thing my advisor, Divy, told after my first paper got rejected. I was obviously very depressed. As a fresh graduate student, I had very high hopes that if I did good work, who would stop it from being published? So Divy gave me a punch line saying that "The PhD is your entire life compressed in 5 years. You'll see ups and downs throughout your life. You'll see ups and downs throughout your PhD". Failures are a part of it. You have to deal with failures. I think, one of the key things I learned during my PhD was to deal with failures. In research, in different parts for getting my research accepted in the broader community, etc. I see that this is very true in other aspects of life as well. This is something that has really helped me get through many situations, helped me understand different scenarios, and overall improve me professionally. That is more from the philosophical point.

From the technical perspective, what I learned during the different projects that I was working on and also during my internship, which I did with Phil Bernstein at Microsoft Research, was the value of having good experimentation for systems research. I even have done this on some my papers as well, and I try not to repeat that. Sometimes we have an idea for a system. We try to do experiments that validate only our idea. As soon as we have that validation, we often write the paper. It is good in some sense to have a validation of the idea, but it doesn't provide others with the insights that they need to maybe apply the idea in a different context. Or maybe learn from what worked or didn't work. So essentially what I try to do now or what I tried to do in some of my later papers in my PhD is to have a more thorough experimental study that analyzes the different tradeoffs of the system. This helps me, as well as others who read the paper, to understand the key insights in addition to what is actually being publicized in the paper. So it helps me understand the system better. It helps me sometimes to optimize the system even more. I hope it helps the readers of the papers to get a better understanding as well. So this is something that I'm sure I'm not there yet. I probably need a lot more practice, a lot more nurturing, a lot more work.

I think a very critical part of doing systems research is to have a good understanding of the dynamics of the system. There are probably hundreds of different ideas that are being proposed in many different papers. But then if we step back and try to make sense of what can

be abstracted out and applied to other contexts, it becomes really hard. That's why I think the VLDB experimental track papers help us a bit in that direction, but I guess the initial papers can also do a better job in that.

Great! Thank you very much for talking to me today.

Thank you! It was a great pleasure. Thank you for having me here.

Data Science and Decision Support at ERIC

Fadila Bentayeb, Julien Velcin, Stephane Bonnevay and Jerome Darmont
Universite de Lyon (Laboratoire ERIC)
Universite Lumiere Lyon 2 – 5 avenue Pierre Mendes-France
69676 Bron Cedex – France
http://eric.univ-lyon2.fr

1. OVERVIEW

ERIC is the French acronym for *Entrepôts, Représentation et Ingénierie des Connaissances*, which literally translates into "Warehouses, Representation and Knowledge Engineering". The ERIC laboratory was created in 1995 and is a joint research unit of two of the three universities in Lyon, both ranked among top universities in France and Europe in their respective fields: Université Claude Bernard Lyon 1, a university of science and medicine, and Université Lumière Lyon 2, a university of humanities and social sciences. ERIC is also a member of the *Institut des Sciences de l'Homme*, a federative institute related to the French National Center for Scientific Research (CNRS).

Research at ERIC aims at extracting value from huge, complex databases, especially (but not exclusively) in the fields of humanities. Our expertise lies in the following domains (Figure 1): 1) data warehousing: intelligent integration of complex data, multidimensional modeling of complex objects, personalized on-line analysis processing (OLAP), data warehouse security; 2) data mining and decision: machine learning, graph study and graph mining, complex data analysis, multicriteria aggregation, opinion mining, data mining software.

ERIC is constituted of two research teams: Decision support Information Systems (SID in French) and Data Mining and Decision (DMD), which address the two aforementioned domains, respectively. Overall, about 50 people regularly work at ERIC, including 6 full professors, 16 associate professors, 25 Ph.D. students and 1 to 3 postdoctoral fellows, depending on the period.

Eventually, the ERIC laboratory is involved in numerous academic and industrial partnerships all around the world, and promotes its research fields by steering and organizing both domestic (e.g., EGC and EDA, both referenced in DBLP) and international events (e.g., the ALT/DS 2012 joint conferences or the VLDB Cloud Intelligence workshop).

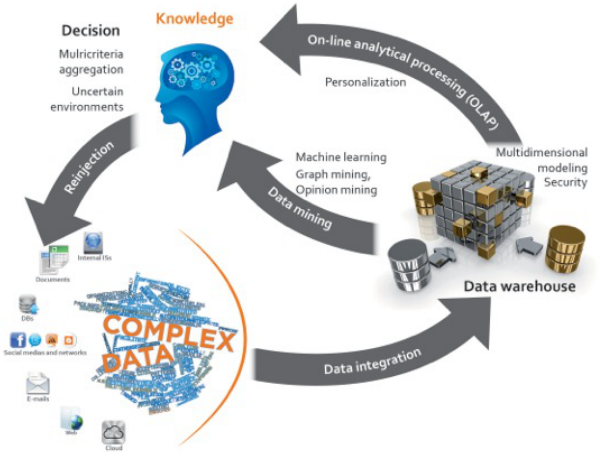

Figure 1: ERIC's Research Topics

In the remainder of this paper, we detail the research topics, some representative results and perspectives of both our research teams.

2. SID TEAM

The expertise of the SID team spans a wide range of areas and applications in data warehousing and OLAP. We aim at developing novel models and methods for designing, storing and analyzing very large-scale data by providing adequate big data warehouses. We address a number of aspects of decision support systems, but mostly focus on the extract-transform-load (ETL) process, multidimensional modeling, data analytics and data/knowledge management infrastructures. We contribute prominently to cloud and big data analytics, by dealing with data from any source, including the Web and social networks.

2.1 Complex Data Warehousing

Traditional systems are very successful in integrating and warehousing structured data for analysis. However, structured data represent only a small

subset of interesting data that could be warehoused by many organizations. Moreover, the classical star schema [12] and its derivatives (snowflake and constellation schemas) are actually relational logical schemas, and prove limited for handling complex data. Thus, we propose methods and tools for integrating and warehousing complex data.

First, to answer to the increasing demand for handling complexity in data sources, we propose an active ETL framework [17] that allows: 1) integrating complex data into an Active XML (AXML) repository [2]; 2) exploiting active rules and mining logged events to self-manage, automate and activate integration tasks.

Our ETL framework is implemented and deployed as a Web application and consists of three main modules: integration services, an event repository, and a management and reactivity module (Figure 2).

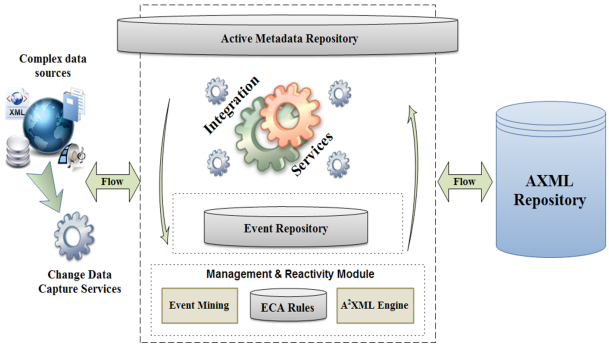

Figure 2: Complex data ETL framework

Integration services input source data and output a set of AXML documents, warehoused in a native XML-based repository. Employing XML and Web services for integrating data helps tackle data heterogeneity, interoperability, distribution and freshness. Moreover, integration services rely on metadata to ensure minimal user intervention for maintaining services. Moreover, services can also be invoked autonomously by triggering Event-Condition-Action (ECA) rules or by AXML calls.

The event repository logs all events related to data sources, integration services and AXML document querying. We apply data mining techniques onto logged events to discover rules [18], which are thereafter used to maintain, automate and reactivate data integration services.

Finally, we achieve system reactivity with a set of active rules that follow the ECA paradigm. ECA rules may be user-defined or automatically mined from event logs. Moreover, invoking embedded services in AXML documents refreshes the repository with up-to-date information. Such embedded services are managed by the AXML engine.

Once complex data are integrated into an AXML repository, which may be viewed as an operational data storage (ODS), we design multidimensional models by exploiting the object-oriented paradigm. We define a layered multidimensional model based on the concept of complex object, which encapsulates data and structure complexity and eases the creation and manipulation of complex data cubes. This model takes full advantage of the object-oriented paradigm to capture multidimensional concepts by symmetrically considering facts and dimensions [4]. In our model, facts and dimensions need indeed not be predefined at the conceptual level, but are designated at analysis time.

Our complex multidimensional model comprises four concepts: complex object, complex relationship, attribute hierarchy and object hierarchy. The package diagram layer models the universe as a set of complex objects, some of which being organized into hierarchies. Complex objects are linked by a set of complex relationships. The class diagram layer provides details about both the structure of each complex object and the origin of complex relationships, which allows defining attribute hierarchies.

This two-layer multidimensional modeling allows users designing complex cubes. To extract complex cubes from the object multidimensional schema, we propose an OLAP operator called cubic projection. Finally, a third layer is constituted of a metamodel for complex cubes that explicitly represents facts and dimensions. Moreover, we provide a translation algorithm that maps any conceptual schema into an XML logical schema, from which an XML physical schema is derived [5]. New analytical functions based on the nature of measure attributes are currently in the pipe.

2.2 Textual OLAP

Recent studies confirm that most data exploited in businesses and administrations are textual, e.g., reports, resumes, e-mails, social data, etc. While OLAP proves very useful for analyzing structured data, it faces several challenges in handling textual data. For example, aggregating numerical data is performed by using standard aggregation functions such as sum, average, etc. Such functions are obviously not suitable to analyze textual data. Thence, to achieve OLAP analyses over textual data, we combine OLAP and information retrieval (IR).

Our key idea is to use a data cube to represent relationships within textual documents. The benefits are twofold: easier representation and processing of text queries, and the creation of new contexts con-

structed by analyzing existing data. We propose a contextual text cube model, called CXT-Cube, which includes several semantic dimensions associated with a specific textual measure [15].

Inspired by IR, we use a space vector model, i.e., an algebraic model that allows representing text documents by a vector of terms in a multidimensional space. Each semantic dimension is extracted from an external knowledge source, i.e., a domain ontology related to the dimension area. Its hierarchy specifies the semantic levels and relationships among text terms in the CXT-Cube. A textual measure is then defined by several vectors of weighted concepts, i.e., one vector per dimension. Term weight is computed with respect to term occurrence frequency and a relevance propagation method, which allows reassigning term scores from leaf nodes to their own ancestors in the concept hierarchy.

One of the possible application domains that would benefit from our textual OLAP approach is recruitment. Better recruitment decisions may indeed be achieved by pre-selecting applicants based on their resumes. A CXT-Cube can be built with respect to resumes (Figure 3) with two semantic dimensions, namely Topic and Location. The Topic dimension includes a topical hierarchy representing domain skills. The Location dimension has three hierarchical attributes City, Region and Country. Another contextual dimension is Time, which represents the sending date of resumes as metadata.

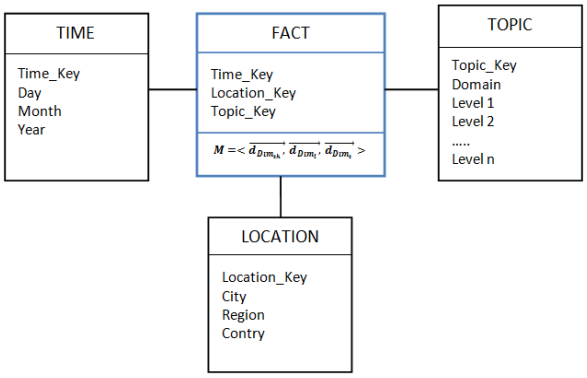

Figure 3: CXT-Cube of resumes

Of course, the main objective of this work is to allow users to easily interact with a CXT-Cube and execute textual queries. To this aim, we propose a new aggregation operator named ORank (OLAP-Rank). ORank's objectives are twofold: 1) aggregating and navigating through textual documents (e.g., resumes); 2) ranking such documents with respect to their relevance degree to a query (Figure 4).

Experiments on resumes highlight the advantage

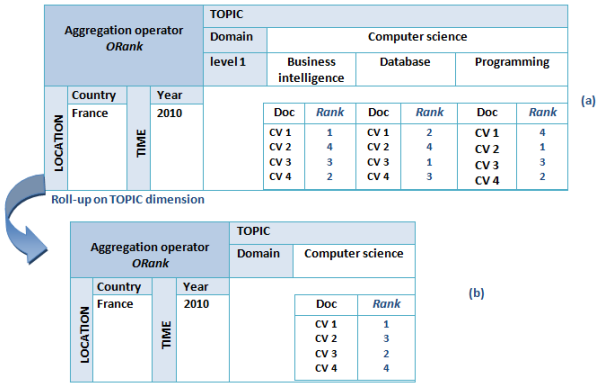

Figure 4: OLAP ranking on resumes

of using ORank over traditional IR ranking functions with respect to recall, precision and precision@k, i.e., the number of relevant documents in the k first retrieved documents divided by k [15].

2.3 Future Research

The advent of cloud computing and the increasing exploitation of big data provide a natural application field to the research performed in the SID team. New opportunities, as well as new research challenges [6], pop up in this environment. We currently aim at developing a cloud computing environment for big data warehousing and OLAP.

Essential issues also remain in parallelizing analytical algorithms for big data to benefit from massively parallel architectures in the cloud. Existing scalable and parallel cloud computing frameworks able to process big data, such as Hadoop, are not adapted to all kinds of processing, e.g., intrinsically interactive processes such as OLAP. However, precomputing expected query answers or aggregates, e.g., by building materialized views or OLAP cubes, can definitely be parallelized to make subsequent visualization and navigation efficient. Thereafter, queries have to be rewritten to run onto distributed chunks of data. In the cloud, with budget coming in as a new constraint in the pay-as-you-go paradigm, view materialization and query rewriting is a scientific lock that is seldom addressed as of today.

On a much smaller scale that we could term "small data", we also start developing a so-called personal intelligence [1] platform called BI4people, to allow very small businesses, organizations or even individuals accessing to simple, cloud-based business intelligence tools.

Finally, as security remains one of the top concerns of cloud users and would-be users, we address data security issues (privacy, availability and integrity) with the help of new secret sharing schemes

for cloud data warehouses and OLAP [3]. However, we still work on demonstrating that the cost of our solution is lower than that of data loss or pilfering, which requires skills in management science.

3. DMD TEAM

DMD team members aim at creating new systems, models and algorithms for data mining and decision making. Complex data come from heterogeneous sources. They are semi-structured, since they can be embedded in a graph-like structure whose nodes and edges are attached to various contents (e.g., text, image, metadata). They are voluminous, imprecise and highly dynamic.

To handle such data, we build on techniques coming from statistics and artificial intelligence: machine learning, information retrieval, multiobjective optimization, multicriteria aggregation, reasoning under uncertainty, etc. Moreover, we claim that producing theoretical results and applying our models to concrete cases are of equal importance. Many applications can be foreseen, such as medical image reconstruction, country ranking, reputation management, social media analysis.

3.1 Structuring Complex Data

In the context of big, interconnected data, the techniques related to data science are especially promising. For instance, DMD team members work on topic modeling for dealing with textual datasets extracted from the Web.

We particularly propose a temporal-aware topic sentiment (TTS) model for dealing with both topic and sentiment over time [7]. Our approach has several important features that are not jointly addressed by other models of the literature, such as ASUM [11] and JST [13]. First, time is jointly modeled with topics and sentiments, which allows capturing the evolution of sentiment about a topic over time. Second, topic-specific sentiments are extracted from the whole data at once, and not from each single document, providing an overall view of topic-sentiment correlations. Finally, no post-processing is needed to match similar topics under different sentiment polarities.

TTS' graphical model is an extension to the classical LDA model (Figure 5). The important point is that sentiment polarity s is drawn given topic z, before drawing both word w and timestamp t given tuple (z, s). We have estimated hidden parameters by classical Monte-Carlo sampling.

We favorably compare our model to other state-of-the-art topic-sentiment models on two datasets extracted from social media: the MDS dataset (Amazon product reviews) and an original dataset of news articles about the Strauss-Kahn case. In particular, we show that our model achieves better results than JST and ASUM for catching the topic-sentiment correlation over time [7].

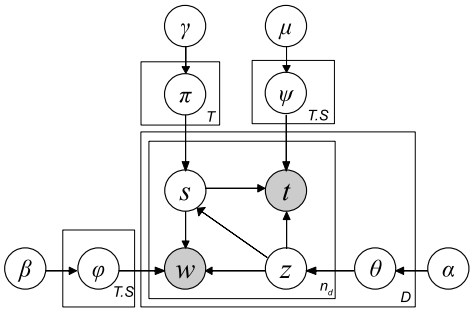

Figure 5: TTS model

We currently try to use this powerful holistic model to improve IR systems by adding new sentiment-oriented features built on TTS.

3.2 Ensemble Methods

Another contribution of the team leverages topological graphs to design metrics that are well-suited to machine learning. Relevant metrics are paramount for solving both unsupervised and supervised machine learning issues.

Obviously, if representing variables are well chosen, similar objects shall belong to the same class or cluster. However, similarity is a relative notion that heavily depends on the density of objects in the description space. This observation leads us to use regions of influence and neighborhood graphs. For instance, in Figure 6, X_{new}'s regions of influence is computed using a relative neighbor graph.

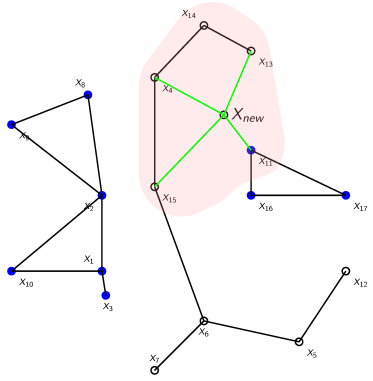

Figure 6: Sample relative neighbor graph

In the domain of supervised inductive learning, we particularly propose an original ensemble method based on neighborhood graphs [19]. This algorithm

constructs several neighborhood graphs by using various projections and, for each of them, infers a label depending on their neighbors' labels. For instance, in Figure 6, this label may be chosen using either the direct neighbors or the two connected components linked to X_{new}.

Then, the different propositions are aggregated to achieve the final result. After conducting an extensive search for the best neighborhood graph and aggregation method, we selected the relative neighborhood graph, which achieves a good balance between too many and too few neighbors.

We compared our approach to state-of-the-art methods on 18 UCI datasets. Our experiments show that we challenge the most powerful techniques, such as Random Forests and SVM. Besides, our approach outperforms classical methods based on neighborhood, such has k-Nearest Neighbors. Our classifier is ranked first when using both the mean error and mean rank of all tested algorithms.

3.3 Multicriteria Decision

DMD team members who work in this field are interested in multicriteria decision, multiobjective decision and collective decision models. They conduct theoretical research to study the properties of various multicriteria analysis methods. Studying the properties of different models is essential to characterize them and understand the pros and cons of one specific model or method over another one.

In particular, we study the generation of Choquet optimal (C-optimal) solutions for biobjective combinatorial optimization problems. C-optimal solutions optimize a Choquet integral. The Choquet integral is used as an aggregation function, presenting different parameters and allowing to take interactions between objectives into account [10].

We propose a new property that characterizes C-optimal solutions [14]. From this property, we define a general method to easily generate optimal solutions in the case of two objectives. We apply our method to two classical biobjective optimization combinatorial optimization problems (BOCO): the biobjective knapsack problem and the biobjective minimum spanning tree problem.

We demonstrate that C-optimal solutions that are not weighted sum optimal (WS-optimal) solutions represent only a small proportion of all C-optimal solutions and are located in a specific area of the objective space, but are much harder to compute than WS-optimal solutions.

This work opens many perspectives. The property we introduce has to be generalized to problems with more than two objectives. It is also interesting to study and define what brings exactly C-optimal solutions that are not WS-optimal. Finally, some specific methods to compute all C-optimal solutions (branch and bound methods) or specific methods to optimize the single objective problems with additional constraints could be studied.

3.4 Future Research

Many exciting challenges lie at the crossroad of DMD members' expertise. To begin with, we continue manipulating graphs for detecting misclassified individuals in supervised learning [16]. Another line of research is to focus on the curse of dimensionality that considerably affects neighborhood graphs. In past studies, we fixed this issue by using random projections that were well-adapted to the general framework of ensemble methods. Now, we are investigating the use of dimension reduction techniques to solve the problem more generally.

In the global context of social media analysis, we recently paid attention to community detection and social role identification issues [8]. It was indeed showed that taking dynamics into account is crucial in this area [9]. A promising track of research consists in addressing both issues jointly, thanks to modern machine learning algorithms based on the description of people, what they write *and* their interactions.

Eventually, according to the multicriteria decision analysis (MCDA) literature, many methods have been developed in the field of multicriteria decision making for eliciting parameters when real preferences are observed. This (quite old) issue is called "preference learning" in machine learning and up to now, there has not been any effective convergence between machine learning and MCDA. One big issue would be to use the great experience of the machine learning community to improve MCDA elicitation methods.

We plan to explore many exciting issues, such as building a collaborative benchmark database in MCDA and studying the possibility of introducing experiment designs in MCDA. Another important issue will be to study how classical MCDA methods, and especially methods based on an outranking principle, can be efficiently used for enhancing machine learning techniques in the big data context.

4. ACKNOWLEDGEMENTS

This paper was written by the laboratory's board, whose members would like to acknowledge the dedication and hard work of all colleagues, Ph.D. students, postdoctoral fellows and administrative staff members. We would also like to thank our external

collaborators all around the world.

5. REFERENCES

[1] A. Abello, J. Darmont, L. Etcheverry, M. Golfarelli, J.-N. Mazon, F. Naumann, T.-B. Pedersen, S. Rizzi, J. Trujillo, P. Vassiliadis, and G. Vossen. Fusion Cubes: Towards Self-Service Business Intelligence. *International Journal of Data Warehousing and Mining*, 9(2):66–88, April-June 2013.

[2] S. Abiteboul, O. Benjelloun, and T. Milo. The Active XML project: an overview. *VLDB Journal*, 17(5):1019–1040, August 2008.

[3] V. Attasena, N. Harbi, and J. Darmont. fVSS: A New Secure and Cost-Efficient Scheme for Cloud Data Warehouses. In *17th ACM International Workshop on Data Warehousing and OLAP (DOLAP 2014), Shangai, China*, pages 81–90, 2014.

[4] D. Boukraâ, O. Boussaid, and F. Bentayeb. Complex Object-Based Multidimensional Modeling and Cube Construction. *Fundamenta Informaticae*, 132(2):203–238, 2014.

[5] D. Boukraâ, O. Boussaïd, F. Bentayeb, and D. E. Zegour. Managing a fragmented XML data cube with Oracle and Timesten. In *15th ACM International Workshop on Data Warehousing and OLAP (DOLAP 2012), Maui, HI, USA*, pages 97–104, 2012.

[6] J. Darmont, T.-B. Pedersen, and M. Middelfart. Cloud Intelligence: What is REALLY New? *1st International Workshop on Cloud Intelligence (Cloud-I 2012), Istanbul, Turkey*, August 2012. Panel.

[7] M. Dermouche, J. Velcin, S. Loudcher, and L. Khouas. A Joint Model for Topic-Sentiment Evolution over Time. In *IEEE International Conference on Data Mining (ICDM 2014), Shenzhen, China*, 2014.

[8] M. Forestier, A. Stavrianou, J. Velcin, and D. A. Zighed. Roles in social networks: Methodologies and research issues. *Web Intelligence and Agent Systems*, 10(1):117–133, 2012.

[9] W. Fu, L. Song, and E. P. Xing. Dynamic mixed membership blockmodel for evolving networks. In *26th International Conference on Machine Learning (ICML 2009), Montreal, QC, Canada*, pages 329–336, 2009.

[10] M. Grabisch and M. Roubens. *Application of the Choquet integral in multicriteria decision making*, pages 348–374. Fuzzy Measures and Integrals – Theory and Applications. Physica Verlag, 2000.

[11] Y. Jo and A. H. Oh. Aspect and sentiment unification model for online review analysis. In *4th ACM International Conference on Web Search and Data Mining (WSDM 2011), Hong Kong, China*, pages 815–824, 2011.

[12] R. Kimball. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. John Wiley, 1996.

[13] C. Lin, Y. He, R. Everson, and S. Ruger. Weakly Supervised Joint Sentiment-Topic Detection from Text. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):1134–1145, 2012.

[14] T. Lust and A. Rolland. 2-additive Choquet optimal solutions in multiobjective optimization problems. In *15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2014), Montpellier, France*, 2014.

[15] L. Oukid, O. Asfari, F. Bentayeb, N. Benblidia, and O. Boussaid. CXT-cube: contextual text cube model and aggregation operator for text OLAP. In *16th ACM International Workshop on Data Warehousing and OLAP (DOLAP 2013), San Francisco, CA, USA*, pages 27–32, 2013.

[16] F. Rico, F. Muhlenbach, D. A. Zighed, and S. Lallich. Comparison of two Topological Approaches to deal with Noisy Labeling. *Neurocomputing*, 2015.

[17] R. Salem, O. Boussaid, and J. Darmont. An Active XML-based Framework for Integrating Complex Data. In *27th Annual ACM Symposium On Applied Computing (SAC 2012), Riva del Garda, Italy*, pages 888–892, 2012.

[18] R. Salem, J. Darmont, and O. Boussaid. Efficient Incremental Breadth-Depth XML Event Mining. In *15th International Database Engineering and Applications Symposium (IDEAS 2011), Lisbon, Portugal*, pages 197–203, 2011.

[19] D. A. Zighed, D. Ezzeddine, and F. Rico. Neighborhood Random Classification. In *Advances in Knowledge Discovery and Data Mining*, volume 7301 of *LNCS*, pages 98–108. Springer, 2012.

Exploring Big Data with Helix: Finding Needles in a Big Haystack

Jason Ellis
IBM Research
jasone@us.ibm.com

Achille Fokoue
IBM Research
achille@us.ibm.com

Oktie Hassanzadeh
IBM Research
hassanzadeh@us.ibm.com

Anastasios Kementsietsidis
IBM Research
tasosk@ca.ibm.com

Kavitha Srinivas
IBM Research
ksrinivs@us.ibm.com

Michael J. Ward
IBM Research
michaeljward@us.ibm.com

ABSTRACT

While much work has focused on efficient processing of Big Data, little work considers how to understand them. In this paper, we describe Helix, a system for guided exploration of Big Data. Helix provides a unified view of sources, ranging from spreadsheets and XML files with no schema, all the way to RDF graphs and relational data with well-defined schemas. Helix users explore these heterogeneous data sources through a combination of keyword searches and navigation of linked web pages that include information about the schemas, as well as data and semantic links within and across sources. At a technical level, the paper describes the research challenges involved in developing Helix, along with a set of real-world usage scenarios and the lessons learned.

1. INTRODUCTION

Enterprises are captivated by the promise of Big Data, to develop new products and services, and to gain new insights into their customers and businesses. But with the promise of Big Data comes an expanded set of problems: *how do enterprises find the data they need for specific purposes*; *how do they make those data usable*; *how do they leverage learning about the data and expertise with the data across tasks*; and *how do they do all these in an efficient manner*. The research community has been quick in responding to enterprise needs but has mostly focused on the problem of efficiency. Indeed, significant effort has focused on providing scalability in the consumption and analysis of terabytes of data [1]. However, there has been little work on addressing in a practical manner the first set of problems, namely, how to help enterprises find *where* all the sources relevant to a task are located; *what* the relationships are between these sources; and *what* data in the sources are relevant to the task and what should be ignored. From our experience, these are the first problems with which customers are coming to us, and only after these problems are addressed it is possible to move to the next step of an efficient analytics solution.

This paper presents Helix, a system we have been actively developing for the last four years that has been used for exploratory analysis of data in various domains. Helix addresses the aforementioned problems by supporting users in iteratively exploring and combining data from multiple heterogeneous sources. Helix users may not have the background, interest, or time to understand all the schemas and data across all sources. However, they all understand what kind of data they are interested in and how they want to use it, so semantic technologies have the potential to bridge from user intention to data representation.

The first technical challenge we address in Helix is bringing together metadata from a variety of models that range across relational, XML, JSON, RDF and even Excel files. We are building a *semantic knowledge base* that describes the entire data corpus. Unlike existing approaches which focus primarily on pairwise integration of schemas within the same model [21], Helix considers multiple schemas across a variety of models. Source schemas may be readily available and can be extracted when sources are added to the system, or are automatically discovered by Helix when sources without schemas are incorporated. Since our approach is geared towards Big Data, even dealing with just the schemas to construct the knowledge base results in scalability issues which need to be addressed [9].

The second technical challenge we address is how to support users in finding relevant data. Keyword search is an obvious starting point, but presenting results from diverse sources differs from presenting document search results. For instance, if a keyword hit is a column name, what would the user like to see? Would other columns in the same ta-

ble be relevant? Or, would sample column values (and values from adjacent columns) suffice? What if the hit occurs in a table row, or in a value of a JSON tree? Do we show the entire row (with possibly hundreds of columns) or the entire JSON tree? There are both strong user experience and data management aspects to this challenge. Currently, Helix guides users in exploring the context in which search results occur using both *schematic* and *semantic* recommendations. The schematic recommendations are consistently presented regardless of the underlying data model. For semantic exploration, we leverage the constructed knowledge base to recommend other data elements, whether in the same or different sources, that may be worth exploring. As users initiate keyword searches and explore data sets through our interface, there is an underlying mechanism that uses the input keyword(s) and user navigational information to build (structured) queries that fetch relevant data from respective sources. In that manner, Helix users can focus on *how to accumulate and filter data of interest rather than focus on how to build queries in various dialects (e.g. SQL, SPARQL, Excel APIs).*

The third technical challenge we address is the discovery of interesting links between diverse sets of instance data. Existing works perform instance link discovery in a *batch* fashion. With huge data sets, and large numbers of sources, these methods generate every possible link, between all possible linkable entities. Generating *all* links not only requires substantial computation time and considerable storage space, but also requires substantial effort since the links must be verified and cleaned. Resources aside, it seems like a huge waste to generate all links only to discover later that actually only a small fraction of them is relevant to most tasks. In Helix, we address the shortcomings of existing approaches by providing a *dynamic* and *context-dependent* linking mechanism. Therefore, we allow the user to specify through metadata which types of links and which data collections she is interested in linking. We then generate at *run-time* links for these data using techniques we have developed for this purpose [15]. When a user has no idea as to what is linkable, we use our automatic identification of linkage points [16] to make recommendations.

The fourth challenge we address is how to assist users in building collections of data elements that are valuable to other users in other tasks. We leverage the internal knowledge base to build a recommendation system for users based on the specific data elements they find interesting. When users choose a specific piece of recommended data, and

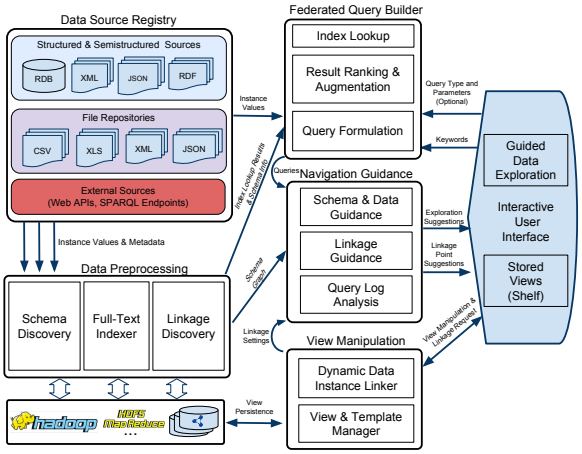

Figure 1: Helix Architecture

treat them as part of the same larger data set, we incrementally build up a high quality set of mappings across users. As more users interact with data, the system builds up knowledge about common portions of the data that get accessed together. This is a bootstrapping approach to building a common semantic model that applies across disparate data silos, and reflects a lower cost to building more formal semantic models/ontologies that can help serve a common view over disparate data.

The overall Helix architecture is shown in Figure 1. We present details of different components of Helix (and respective challenges) by following logically the way the system is used in practice. So, in Section 2, we describe the pre-processing phase in Helix which includes schema discovery, linking, and indexing of input data sources. Section 3 presents the guided data exploration mechanism, while Section 4 presents several usage scenarios. Section 5 discusses related work, and Section 6 concludes the paper with a summary and a few interesting directions for future work.

2. PRE-PROCESSING PHASE

All input data sources in Helix are defined in the data source registry component. There are three classes of sources considered. The first class includes (semi-)structured sources with predefined schemas and query APIs, such as relational databases and triplestores. The second class is (online or local) file repositories, such as the ones published by governments (e.g., data.gov or data.gov.uk, or data sources published by U.S. National Library of Medicine), or in cloud-based file repositories (e.g., Amazon S3). Finally, the third class are those sources directly read from online Web APIs, e.g., data read using the Freebase API.

Figure 2: Example JSON Data

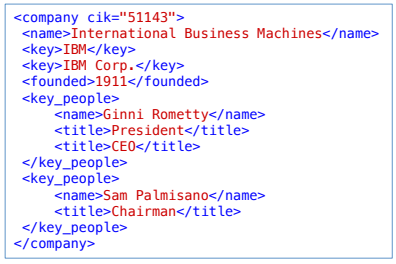

Figure 3: Example XML Data

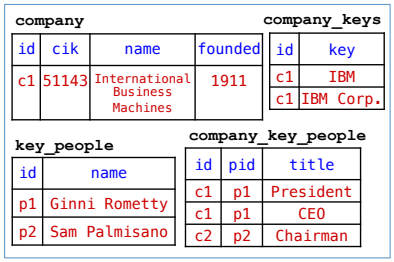

Figure 4: Example Relational Data

One of the goals in Helix is to process data based on explicit user needs and avoid unnecessary or expensive pre-processing given that we are dealing with Big Data. Therefore, the data pre-processing phase comprises only three *essential* steps, all performed in a highly scalable fashion implemented in the Hadoop ecosystem: (a) schema discovery, where each input source schema is represented in a common model in the form of a *local schema graph*; (b) full-text indexing, where data values and source metadata are indexed; and (c) linkage discovery, that incorporates instance-based matching and clustering of the (discovered) schemas. The outcome of the pre-processing phase is a *Global Schema Graph* which plays a key role in Helix. In the following, we discuss briefly each of the steps and how the Global Schema Graph is constructed.

2.1 Schema Discovery

The schema discovery process begins by constructing a *local schema graph* for each of the input sources. Intuitively, schema graphs are used as the common schema representation format that alleviates the differences across the models of different sources. We distinguish two types of nodes in the schema graphs: (a) *attributes*, which across models correspond to schema elements whose domain is literal values (e.g., column names in the relational model, PCDATA elements in XML, strings in JSON, etc); and *types*, which across models correspond to schema elements whose domain is defined (recursively) through other type or attribute nodes (e.g., tables or schemas in the relational model, intermediate elements in XML trees, etc.).

Given the wide range of sources considered in terms of data models, the local schema graph construction is customized to each data model. In more detail, for semistructured data with no provided schema, we first build a minimal schema graph (more precisely a tree) [17] that contains all the possible paths in the semistructured data instances. That is, nodes in the constructed local schema graph correspond to elements of the semistructured data, and a path in the schema graph

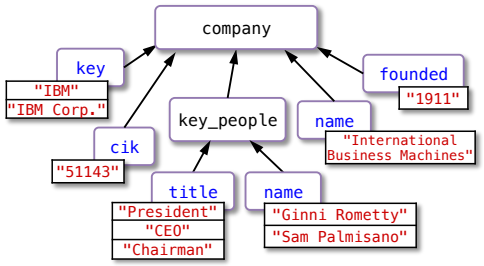

Figure 5: Example Local Schema Graph

exists if there is at least one data instance having that path. To illustrate, Figures 2 and 3 are two examples that show essentially the same data in both JSON and XML formats. Figure 5 shows a schema graph that can be extracted from either of them. In the graph, `company` and `key_people` are types, while `name` is an attribute. Our local schema graph construction is similar to that of approximate DataGuides [10] or representative objects [20]. Since our goal is a simple and concise representation of the structure of the data, we do not employ techniques that try to derive an accurate conceptual model of the input data. Instead, we use techniques that provide a schema that initially facilitates users' understanding of data, while providing ways to incrementally build more complex and accurate models through the user interaction and input.

For (semi-)structured data with a provided schema (e.g., through available metadata or DDLs), the schema graph can be constructed without analyzing instance values. So, for the relational source in Figure 4, the schema graph of Figure 5 can be constructed by using the table definitions and foreign key relationships. If schema definitions including foreign key information are not available, one type is created per table with table columns again becoming attributes of the type, and a foreign key discovery process [23] is performed along with link discovery (see Section 2.3) to connect the different tables.

For RDF (graph) data, each class is represented with a type, and its properties that have literal values become type attributes. Properties that link

two classes in RDF result in links between the corresponding types. Here again, if an RDF Schema is given, then it is used to construct the local schema graph. In the absence of an RDF schema, Helix infers the schema graph by going through all the instance data (RDF triples) and figuring out the resources, their classes (i.e., the `rdf:type` statements), their properties, and the properties between classes.

Once the schema graph for each source is constructed, a Uniform Resource Identifier (URI) is assigned to each node. The assignment is such that it not only captures the relative position of each node within the schema graph, but also captures the provenance of each node, e.g., the name of the source as well as its model. As an example, assume the schema graph in Fig. 5 is derived from a JSON source with name `compdata`. The URI for the property `cik` of object `company` is `json://compdata/company/cik`. Similarly, if the relational source in Fig. 4 is registered with name `cmpdt` and the tables are stored in a schema named `db2usr1`, `rdb://cmpdt/db2usr1/company/name` is the URI of the node corresponding to column `name` in table `company`.

2.2 Full-Text Index

In Helix, each instance value is also assigned a URI that can be used to locate that specific value. Figure 5 shows the instance values associated with each attribute in our example. We view instance values as documents and use a standard Information Retrieval (IR) engine to tokenize and index the instance values to allow standard keyword search across the input data sources with Boolean queries and fuzzy search capabilities. There are several ways to model instance values as documents:

- Treat each instance value as a document. So each instance value is assigned a URI that can be used to locate it. For example, URI `json://compdata/company[0]/key_people[0]/name[0]` is associated with "Ginni Rometty" in Figure 2.
- Treat all instance values of an attribute as a single document, with the attribute URI as the document identifier. For example, values "IBM" and "IBM Corp." in Figure 2 can form a document associated with URI `json://compdata/company/key`.
- Group values that appear together in one instance of a type and treat them as a single document. This results in *row-level* indexing for relational data, node-based grouping of instance values for tree-structured (e.g., XML) data, and object-level indexing for graph (RDF) data. In Figure 2, this approach results in one index entry for the `company` type instance, and two entries for `key_people` instances.

We support all the above indexing mechanisms as options, and our experience shows that the first approach is superior to others, with respect to keyword search during guided exploration (see Section 3). However, a particular challenge with this approach is the size of the index, given the number of instance values and the fact that instance URIs are long strings. As a result, we have devised a simple compression mechanism to index each *distinct* value only once for each attribute. This does not impact guided exploration, whereas it results in much smaller index size in practice. Regardless of the indexing mechanism, the attribute URIs are also stored in the index to facilitate attribute-based grouping of the keyword search results. Unlike previous work on the indexing of heterogeneous data [8] which are limited to the indexing of values, our index is further extended with indexing of metadata, i.e., with the types and attribute names themselves to provide a unified keyword search index.

2.3 Linkage Discovery

The last phase in pre-processing is discovering links between different types and attributes *within* as well as *across* the schema graphs of different sources. Traditional schema-based matching is not effective in matching highly diverse and automatically-constructed schemas where the labels of schema elements are not always representative of their contents, and data come from several sources that use different models and representations. Therefore, our approach is to perform an all-to-all instance-based matching of all the attributes. Scaling the matching process for a large number of attributes and large number of instances per attribute is a major challenge. We address this problem by casting it into the problem of computing document similarity in information retrieval [9]. Specifically, we treat each attribute node as a document, and we consider the instance values for that attribute as the set of terms in the document. To scale the computation of pairwise attribute similarity, we use Locality Sensitive Hashing (LSH) techniques, as is done in computing document similarity. Briefly, we construct a fixed small number of *signature* values per attribute, based on MinHash [3] or Random Hyperplane [4], in a way that a high similarity between the set of signatures guarantees high similarity among instance values. This results in efficient comparison of instance values between attributes. We then create small buckets of attributes so that similar attributes are guaranteed to be in the same bucket with a high probability. This is similar to a common indexing technique used in

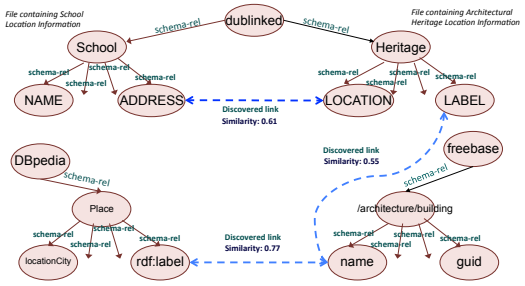

Figure 6: Sample Schema Graph

record linkage known as *blocking* [5]. Our experiments on large data sources show that our approach is very effective in reducing the number of pairwise attribute comparisons required for an all-to-all attribute matching [9].

In our evaluation, we found that the precision and recall of linkages between attributes with textual values is very good [9]. However, linkages between attributes with numeric or date/time values tend to have little semantic value, even when the similarity of their instances is high. Currently, we optionally filter attributes with these data types. We are investigating the scalability of *constraint-based instance matching* [21] for discovering linkages between such attributes.

The attribute-level linkages found within and across data sources are used not only for guided navigation of the sources (see Section 3), but also to find type-level linkages and grouping (clustering) of types. In more detail, type clustering is performed to group types that have the same or highly similar attribute sets. For example, all 'address' types of an XML source might create a single cluster, in spite of these addresses appearing at different levels and under different elements of the tree. Type-level linkages induce a similarity graph, where each node represents a type and the weight of an edge connecting two types reflects their similarity. This similarity is the average of (a) the instance-based similarity between the attributes of the two types; and (b) the Jaccard similarity between the sets of attribute labels of the types. An unconstrained graph clustering algorithm [13] is then used to find clusters of types in the similarity graph.

2.4 Global Schema Graph

The schema graphs of all the input sources along with discovered attribute and type linkages are all used to build the *Global Schema Graph*. This graph provides a unified view over the input sources, enables navigation, and allows the discovery of related attributes and types through schema and similarity-based linkages. Figure 6 shows a portion of a global schema graph constructed for one of our use cases. In this example, a data set on national heritage sites in the city of Dublin is linked to a data set in the same source containing school locations, based on the similarity of the address/location attributes in the two data sets. The data set is also linked to a type in a Web knowledge base that contains information on architectural buildings, which itself is linked to another knowledge base containing information about public locations (*Place* type in an ontology). These links implicitly show that these data sets contain information about locations, and that there is potentially a connection between school locations and national heritage sites in the city of Dublin, one of many exploration capabilities of Helix. In the figure, we distinguish two sorts of links, namely *explicit* links (drawn in solid black lines) that are inferred by looking at individual sources through schema discovery (see Section 2.1), and *discovered* links (drawn in dashed blue lines) that require additional logic and consider multiple sources (see Section 2.3). For discovered links, we add annotations to capture their computed similarity, as well as the method by which the link was discovered (e.g., MinHash, user generated, etc.).

The global schema graph is a key structure in Helix since it governs and guides user interactions (more details in Section 3). What is less obvious though is that there are technical challenges in terms of managing the graph itself. Helix is geared towards Big Data scenarios, and as more and more sources are incorporated into the system, the global schema graph very quickly becomes quite large. As the system continuously queries, updates, and augments the graph, it is important that all these operations are performed efficiently; otherwise the global schema graph ends up being a bottleneck to the system performance. To address these challenges, we store the global schema graph in our own graph store, called DB2RDF, which has been proven to outperform competing graph stores in a variety of query workloads using both synthetic and real data [2]. Our graph store supports the SPARQL 1.0 graph query language [24] and interactions with the global query graph are automatically and internally translated to SPARQL queries.

3. GUIDED EXPLORATION

Once a global schema graph is built, one might think a user could somehow use the global schema graph directly to query the data and navigate through related pieces of information. However, despite several efforts on our part to help users directly explore the schema graph (our first prototype followed this approach [14]), comprehending

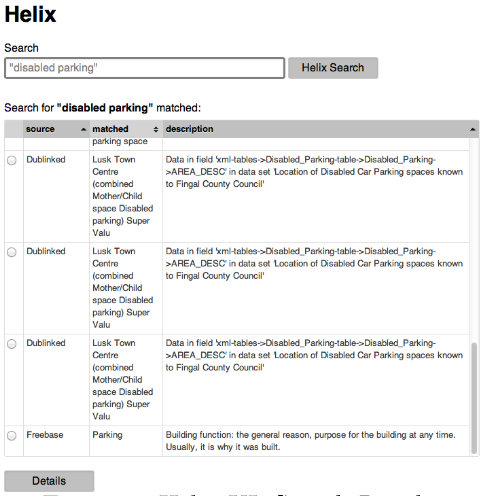

Figure 7: Helix UI: Search Results

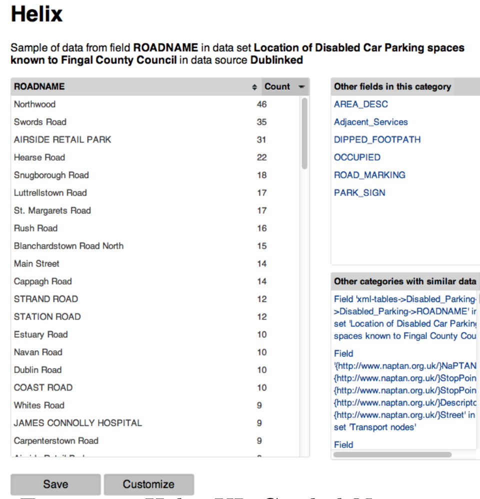

Figure 8: Helix UI: Guided Navigation

the myriad of schema elements and their connections turned out to be too much for users to grasp. We also tried using keyword searches directly on the global schema graph to help users construct structured queries as in [26], but this technique did not help the construction of complex queries required by some of our use case scenarios (Section 4). We have thus arrived at the approach of guided data exploration, which allows users to construct complex queries *iteratively* by building on some basic building blocks. Guided exploration in Helix has four components: (1) keyword search, (2) guided navigation, (3) construction of virtual views, and (4) integration across virtual views to build more complex queries.

3.1 Keyword Search

Users initiate their guided exploration with a keyword search over the index described in Section 2.2. Our search engine results (see Figure 7) are customized in such a manner that the result set contains not only the hits from the global schema graph for the input keyword(s) (with each hit being either a type or an attribute hit, and shown in the column labeled "matched" in Figure 7), but also the name of the data source in which each hit appears (column "source") as well as the precise location of the hit within the source (column "description"). As with any search, the search results are rank ordered, but they can be sorted by values in any of the columns. The search process is, by itself, not novel compared to those found in the literature. We use it simply to initiate an exploration in the system.

3.2 Guided Navigation

Guided data navigation is an iterative process that assists users in confirming that data that they are viewing are relevant to their tasks and in discovering closely related data elements. Users initiate navigation by choosing a hit on the search engine results page. Helix displays a screen with three primary components, a table containing sample data for the chosen hit and two lists of links to related attributes and types (see Figure 8). Clicking on one of these links takes the user to a new data navigation screen, this time populated with data and links for the attribute or type clicked on. The presentation is uniform regardless of the data model of the underlying data source. The user is guided to structurally and semantically related attributes and types, reinforced at each step with sample data. She can focus on the data itself rather than how it is represented and accessed. We describe the main features of the guided navigation interface next.

- Data sample: A sample of popular instance values are presented when a user drills down on a hit on an attribute. The objective is to help users decide if the selected hit is relevant to their task. If the hit is on a type, it is difficult to determine which of the attributes for the type should be displayed. In this case, the user can explore data associated with that type using the schema links described next.
- Schema links: Helix displays links found during the schema discovery process (see Section 2.1) for two reasons. First, schema links inform users about *other* attributes and types in the *vicinity* of the hit (i.e., in the same source) and may be relevant for their task. Second, we have observed that this sort of navigation often guides users to the right data even if the original hit was not on the right attribute. The list of schema links are navigable, and ordered by the number of instances in the attribute.
- Discovered links: For a given attribute or type

hit, we show links to other attributes or types that were discovered during pre-processing (see Section 2.3). These links are similar to recommendations for relevant data. Our evaluation of these links over Web-based data sources [9] indicates that the precision and recall over enterprise data sources is high as well. This list is ordered initially by the similarity score calculated during pre-processing. A user validates a discovered link indirectly by clicking on it and using the relevant data from it (as described in Section 3.3). When this occurs, we boost the link between the two types or attributes (the one in the hit and the one the user navigated to from the hit) to the maximal similarity value of 1.0, and we annotate the link to indicate that it is user validated. These links are subsequently ranked higher when users again browse the same hits and their associated links.

We have observed that the simple navigation techniques described here often help users find relevant data after a few navigation steps. Occasionally a hit is exactly what the user is looking for; at other times, the data of real interest are for a different attribute or type in the same source; and sometimes one of the discovered links guides the user to more relevant data in another source. The key message here is that, in practice, users rarely find what the are looking for in *one shot*. This is even the case in web searches. What is important is that Helix provides a whole infrastructure to help users *zero in* on the data they want to use. Our experience from different usage scenarios shows that this process is typically faster than using a pure-text search-based approach that lacks both the context of the hits and the connections with other related areas of the search space.

3.3 Virtual View Construction

When a user has found an interesting type, she can construct a virtual view on that type, and save it on a *data shelf*. The steps to accomplish the creation of a virtual view involve customizing the type on the guided navigation screen. At the interface level, the user chooses various attributes of a type in tabular form, and is never aware of the actual data format. The user actions available in this step include simple projections, filtering and ordering on any of the attributes of a selected type (see Figure 9). In the back end, because our internal representation of the global schema is a large graph, a 'virtual' view corresponds to a Basic Graph Pattern (BGP) in SPARQL, which (in its most basic form) starts with the template `?x type <T>`, where

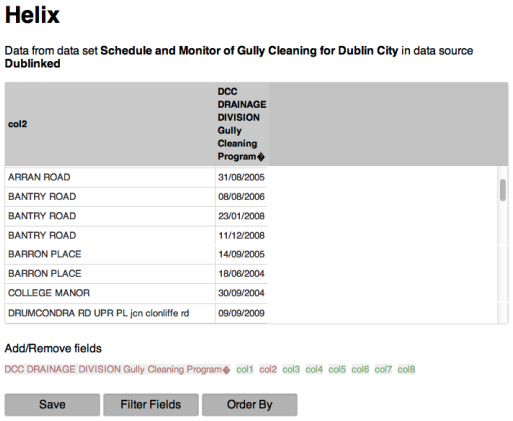

Figure 9: Helix UI: Virtual View Construction

`<T>` is a type, and adds statements of the form `?x <P> ?y`, as the user projects attribute `P`. Note that at the interface level, the user is not required to be aware of any formal structured query language. We build up the structured query as the user interacts with a simplified version of the data in its tabular form. Filtering and ordering are likewise internally represented using equivalent SPARQL operators.

Once a virtual view has been constructed, the user can save it on a *data shelf* which just saves a SPARQL query in the user's profile. The user can also annotate the virtual view with terms that more accurately describe it. These terms are also added to the full text index (where the 'document' being indexed is now a query), such that subsequent searches in guided exploration can lead to hits on virtual views as well as original datasets. As stated in the prior section, we consider saving a virtual view an indication of the usefulness of the guided navigation step and boost the similarity values of discovered links followed during the construction of the virtual view.

3.4 Integration of Virtual Views

The real power of Helix is that these simple virtual views can now be used as building blocks for more complex views. To build such views, we provide two operators which correspond loosely to a JOIN and a UNION operation. The difference between a standard JOIN operation and our notion of JOIN is that our version is actually a *semantic* JOIN that corresponds to an instance linkage operation between the instance data of different virtual views. The UNION operation is more straightforward as it has the usual semantics. Two issues require further explanation. First, we need to explain how we actually fetch that data for the virtual views that are to be joined (or unioned). Second, we need to explain how the actual join occurs.

In Helix, because the data sources being joined may not even be in the same format, and not all of the formats have uniform query engines (or even have an engine), we use the global schema graph as a semantic model that unifies distributed data sources with *direct schema mappings* from the global schema to each of the data sources schema. Differences in query engines (and languages) are then accommodated by using the global schema to local source schema mappings and knowledge of the query language/API associated with the underlying source to translate the view specification in SPARQL to an appropriate query (e.g., SQL for relational, XPath for XML) that is executed over the underlying source to fetch the instance data. Note that for data formats that do not support a query engine (like CSV or Excel), we house the data in a key-value store to account for variable schema across multiple datasets.

With the instance data in place, we need to perform the actual JOIN operation. Unless specified by the user, Helix recommends a set of attribute pairs on which the views may be linked. This recommendation is not purely based on the links found during the linkage discovery step described in Section 2.3, since attribute and type similarity does not always imply that instance-level linkages exist. For instance, two types may be linked because of similarities at the term level, but actual instance values might differ (e.g., *Bank of America* versus *Apple Bank*). Our recommendation of pairs of attributes for linkage is based on our work on automatic identification of linkage points [16]. Briefly, we recommend a pair of attributes as a linkage point if they can effectively be used to link instances of the two types. The linkage points are ranked based on their *strength* [16], which reflects the ability to establish a reasonable number of one-to-one links between the instances. Note that we consider all the attributes of the type in the view, not just those explicit in the stored query. The user can then choose a pair (or pairs) of the recommended attributes. Helix invokes its Dynamic Data Instance Linker (see Figure 1) that uses fuzzy matching techniques [15] to link the views and presents the results to the user, together with a confidence score (see Figure 10). The user can accept the results by saving them to the shelf or go back and choose different or additional linkage points and try again. When a new view is saved, internally the SPARQL algebra describing the view also records the selected linkage points. As an aside, since each view is a BGP in SPARQL, linkage points might occur between variables that don't necessarily have the same name in the SPARQL. The linkage is therefore expressed as a filter that calls a JOIN operation on the two variables that are being joined.

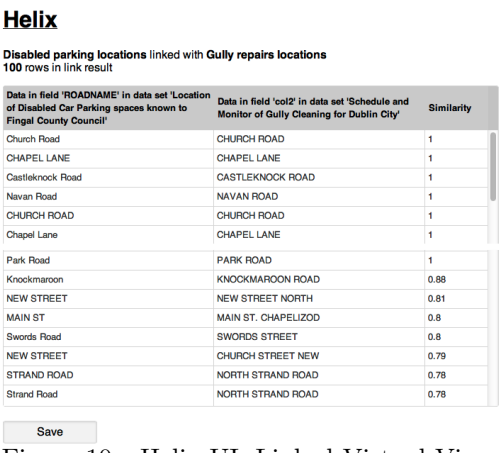

Figure 10: Helix UI: Linked Virtual Views

4. USAGE SCENARIOS

The design and implementation of the Helix system has gone through extensive evaluation using several usage scenarios in different domains. The majority of the usage scenarios are inspired by our interactions with customers, in trying to understand their needs in data exploration and help them with the first steps of their data analytics tasks. In this section, we describe two such usage scenarios and some of our key observations and lessons learned. We first describe details of usage scenarios using data published by the city of Dublin, Ireland. Extracting relevant information from online public data repositories such as those published by government agencies is a frequent request within enterprises. We then describe a customer relationship management (CRM) use case as an example enterprise data exploration scenario. Finally we share some of the lessons learned through these and other applications of Helix. Note that our goal here is not to perform a scientific study of the effectiveness of the algorithms implemented in the system (such as the study we have performed on accuracy of attribute-level linkages [9] and linkage point discovery [16]). Nor do we intend to evaluate the effectiveness of our user interface through a large-scale user study, which is a topic of future work and beyond the scope of this paper.

Table 1 provides a summary of the source characteristics in the two scenarios, and Table 2 provides the total number of links found across these sources. Each source is in itself composed of multiple data sets. We therefore provide a summary of the number of links between data sources, as well as the

Table 1: Summary of Data Sources

Data Source	Types	Instances	Tables/files
Bug reports	201	7M	1
Bug fixes	95	121M	7
Freebase	1,069	44M	NA
DBpedia	155	2M	NA
Dublinked	1,569	22M	485

Table 2: Links Across Data Sources' Types

Data Src/Data Src	#Links	Data Src/Data Src	#Links
Bug fixes/Bug fixes	1,510	Bug reports/Freebase	298
Bug fixes/Bug reports	1,209	Bug reports/Bug reports	316
Bug fixes/DBpedia	25	Dublinked/Dublinked	288,045
Bug fixes/Freebase	1,216	Dublinked/DBpedia	225
Bug reports/DBpedia	4	Dublinked/Freebase	2,351

summary of links within a single data source (e.g., a single data source like Dublinked is composed of several hundred files). The number of links is provided to demonstrate that the system computes a large number of them. It is not our intent here to characterize them by the standard metrics of precision and recall (cf. [9]). As the links are used primarily within the context of a rather focused search, we illustrate in the use cases below how sample links may help data discovery and analysis.

4.1 Dublinked

The city of Dublin has a set of data from different government agencies that is published in a number of different file formats (see: `http://dublinked.ie/`). At the time of this writing, Helix could access 203 collections. Each collection consists of multiple files, resulting in 501 files with supported formats that broke down into 206 XLS, 148 CSV, 90 DBF, and 57 XML files. Helix indexed and pre-processed 485 files, but 16 files could not be indexed due to parsing errors. Our main use case here is data integration across the different agency data, but we also decided to connect the Dublinked data to Freebase and DBpedia, to determine if we could use the latter two sources as some form of generic knowledge. For the pre-processing step, we processed DBpedia and Freebase as RDF dumps.

The value of integrating information across files and across government agencies is obvious, but we illustrate here a few examples, based on links discovered in our pre-processing step, in Table 3. Here are some examples of questions that a city official can now construct queries for, based on Helix discovered linkages in the data shown in the table:

1. Find schools that are polling stations, so that the city can prepare for extra traffic at schools during voting periods.
2. Find disabled parking stations that will be affected by pending gully repairs, to ensure accessibility will be maintained in a specific region of the city.
3. Find recycling stations that handle both cans and glass to route waste materials to the right stations.

Table 3: Sample Links for the Dublinked Scenario

Property Pairs	Score
`xml://School-Enrollment/Short-Name` → `xml://Polling-Stations-table/Name`	0.82
`csv://DisabledParkingBays/Street` → `csv://GullyRepairsPending/col2`	0.68
`xls://CanRecycling/col0` → `xls://GlassRecycling/col0`	0.71
`csv://PostersPermissionsOnPoles/Org` → `csv://CandidatesforElection2004/col2`	0.54
`csv://CandidatesforElection2004/col1` → `csv:/CandidatesforLocalElection2009/col5`	0.97
`csv://PlayingPitches/FACILITY-NAME` → `csv://PlayAreas/Name`	0.40
`csv://FingalNIAHSurvey/NAME` → `http://rdf.freebase.com/architecture/structure/name`	0.56
`dbf://Nature-Development-Areas/NAME` → `http://rdf.freebase.com/sports/golf-course/name`	0.55
`csv://ProtectedStructures/StructureName` → `http://dbpedia.org/HistoricPlace/label`	0.42

Table 4: Type Clusters for the Dublinked Scenario

Type Clusters

`xml://SchoolEnrollment20092010-1304`
`csv://SchoolEnrollment20102011-2139`
`xml://SchoolEnrollment20102011-2146`
`csv://SchoolEnrollment20082009-1301`
`xml://Schoolenrollment20082009-1303`

`csv://GullyCleaningDaily2004-11CENTRALAREA-1517`
`csv://GullyCleaningDaily2004-11NORTHCENTRALAREA-1518`
`csv://GullyCleaningDaily2004-11NORTHWESTAREA-1518`
`csv://GullyCleaningDaily2004-11SOUTHEASTAREA-1519`

4. Find organizations who have the most number of permissions to put posters on poles, to assess organizations with maximal reach to citizens.

In general, links alert users to the possibility of related data that could be pooled before any analytics is performed. For instance, any analytics on play areas would likely need to include the data in PlayAreas file as well as the Play pitches file. Similarly, time series analysis of election data would likely include the 2004 file as well as the 2009 file. Finally, links to external data sets can easily imbue the data with broader semantics. As examples, the Name column in the FingalNIAHSurvey file refers to architectural structures, but another column also called 'Name' in the Nature-Development-Areas file is really about golf courses or play areas. Similarly, the StructuredName column in the ProtectedStructures file is about historic structures.

Table 4 shows two sample type clusters that Helix discovered in the Dublinked data. Recall that type clusters are based on similarity of the schema elements in the type, as well as the instance similarity of each of those elements. The first cluster (files starting with `SchoolEnrollment*`) groups data by year despite changes in the data format. The second cluster (files starting with `GullyCleaningDaily2004*`) discovered by Helix groups data by area, as is apparent from the titles of the files. Following our design goals in Helix, the system itself is not trying to interpret the semantics of each discovered cluster. It will provide a tool for a knowledge worker to specify data sets for meta-analysis.

4.2 CRM

In most enterprises, maintaining a consistent view of customers is key for customer relationship management (CRM). This task is made difficult by the fact that the notion of a customer frequently changes with business conditions. For instance, if an enterprise has a customer *"IBM"* and also a customer *"SoftLayer"*, they are distinct entities up until the point that one acquires the other. After the acquisition, the two resolve to the same entity. The process of keeping these entities resolved and up to date in the real world is often a laborious manual process, which involves looking up mergers and acquisitions on sites like Wikipedia and then creating scripts to unify the companies involved. Our second scenario targets this use case. The real world sources involved are (a) a relational database with a single table that tracks defects against products (b) a relational database that tracks fixes for the defects in the defect tracking database (with 7 tables – one per product), (c) an RDF version of Wikipedia, from Freebase/DBpedia.

The query that the knowledge worker is interested in is a picture of the number of defects fixed for each customer (where each customer is grouped or resolved by merger and acquisition data). We highlight the features of Helix that help the user build this query. We illustrate what steps a user would take in Helix if her intent is to build a table of customer records of bugs and their corresponding fixes, accounting for the latest mergers and acquisitions. Note that because much of this data is proprietary, we do not display confidential results.

Step 1 The user issues a keyword search on a customer name, such as 'IBM', to see what they can find. The hits returned include records in the bug database, as well as nodes in the Freebase/DBpedia RDF graph which match IBM (e.g., IBM, IBM AIX, etc). The user then clicks a particular hit in the bug reports database to explore the record in the context of the original table/graph. The user sees the larger context for the table (other records in the column that contains IBM, and other columns in the table that are related to the `CUST-NAME` column within the same table). More importantly, the user finds other properties that also contain similar data (as an example, see Table 5 that shows some real links found by Helix). If the user browses the `CUST-NAME` column in the bug reports database, Helix recommends the `CUSTOMER` column in the bug fixes database, and the `/business/organization` type in the RDF Freebase graph based on the links.

Step 2 The next step is the creation by the user of multiple virtual views that are placed on the data shelf (see Figure 11). Step 2 is a direct outcome of the data exploration conducted by the user in Step 1, where the user finds relevant data, and now wants to subset it for their task. For this example, we assume the user creates 3 virtual views. The first view contains a subset of bug reporting data with the columns `CUSTOMER` and `BUG NUMBER`, the second contains a subset of the bug fixes data with the columns `BUG NO`, `FIX NO`, and the third contains a subset of Freebase data, with `/business/employer`, and its relationship to its acquisitions through the `/organization/companiesAcquired` attribute.

Table 5: Sample Links for the CRM scenario

Property Pairs	Score
`rdb://Bug-Reports/CUST-NAME` → `rdb://Bug-Fixes/Product1/CUSTOMER-NAME`	0.75
`rdb://Bug-Reports/CUST-NAME` → `rdb://Bug-Fixes/Product2/CUSTOMER-NAME`	0.74
`rdb://Bug-Reports/CUST-NAME` → `http://rdf.freebase.com/business-operation/name`	0.47
`rdb://Bug-reports/CUST-NAME` → `http://dbpedia.org/Company/label`	0.28

Step 3 This step involves using semantic joins to build more complex views customized for the user's task. Here, the user likely joins Views 1 and 2 on `BUG NUMBER` and `BUG NO` to create View 4 of bugs that were fixed for different customers. Then, the user joins Views 3 and 4 on `CUSTOMER` and `/organization/companiesAcquired` to create View 5 of bugs and fixes by customer, where the customer record also reflects any companies acquired by customers in the bug report/fixing sources. At this point, the user could union View 4 with View 5 to find a count of bugs and fixes delivered to a customer and any of its acquisitions. Figure 11 shows all the steps in the process. In the figure, notice that a bug like 210 which normally would only be attributed to customer *"SoftLayer"* is now also counted as part of the bugs for customer *"IBM"* since the latter acquired the former. Knowledge of these acquisitions can be used to further *refine* the result by, say, removing all *"SoftLayer"* entries since they are already considered as part of *"IBM"*.

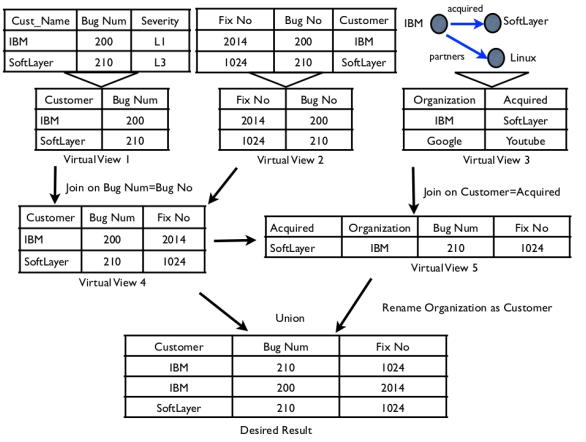

Figure 11: Steps in the CRM Scenario

4.3 Other Use Cases

In addition to the two use cases described above, we have used Helix in a number of other domains and use cases. We have found unexpected correlations among data sets that require examining instance linkages. An example is discovering a rather strong connection from company names whose products have been recalled to names of companies that make Peanut Butter products, compared to say, those that make H1N1 products or Vegetable Protein products. This scenario uses public data published by the U.S. Department of Health and Human Services. In such scenarios, the instance linkages performed through the user interface are examined to measure the relative importance of the discovered links. Another discovery is the use of unexpected labels, or changes to the type of data instances stored without a change in the schema. For example, in a scenario involving air quality data from various government agencies, we observed schema elements labeled as "zip" that contained string-valued addresses. Note that all these scenarios have been performed using a system that can be set up by non-expert users who have little or no technical background on data management technologies; they only need to specify data access mechanism for input data sources.

5. RELATED WORK

Our work builds upon and extends techniques from several research areas including data management, information retrieval, semantic web, and user interface design. In terms of the overall system, Helix can be seen as a novel DataSpace Support Platform (DSSP). It offers many of the features envisioned for DSSPs by Halevy et al. [12] including dealing with data and applications in a wide variety of formats, providing keyword search over heterogeneous sources, and providing "tools and pathways to create tighter integration of data in the space as necessary". To our knowledge, Helix is the first such system that allows generic "pay-as-you-go" integration mechanism that works on heterogeneous data across a wide variety of domains and applications.

Previous work has proposed systems that perform analysis and "pay-as-you-go" integration in specific domains. Kite [22] supports keyword search over multiple unrelated relational databases, but semantic integration is achieved using foreign key discovery techniques that are not well-adapted to tree- and graph-based data sources, and it does not scale well to a large number of databases. iTrails [27] accommodates semi-structured data sources, and its internal data model is similar to Helix's local schema graphs. Its integration semantics are provided through *trails*, query rewriting rules specified in an XPath-like language. These trails are written by the system provider, though the authors propose methods for automated generation. The Q system [25] has several features similar our system: it constructs a schema graph over structured and semi-structured data sources, with edges representing both structural and semantic relationships, including ones discovered through schema matching; and its users provide feedback to improve query results. Q's domain is primarily scientific data sources that are reasonably well-curated. It generates queries from keywords by computing top-k tree matches over the schema graph, then displays results directly to users, while Helix utilizes an iterative, link-following approach to incrementally build results. With Q, users provide explicit feedback on results based on both the answers and the queries (trees) that produced the answers, requiring a fair degree of technical sophistication; our system derives implicit feedback based on what users do with results. Google Fusion Tables [11] emphasizes data sharing, visualization, and collaboration on Web-based tabular data sources, with only simple data integration support. QuerioCity [18] is designed to catalog, index, and query city government data. Its approach is similar to our support of semistructured file repositories, although the specific domain allows certain tasks to be further automated (e.g., automatic linkage to DBpedia entities and types).

Our work on the user interface is related to research on providing non-expert users with search and query capability over standard database management systems. In particular, our goal is providing an *exploratory search* [19] mechanism over large heterogeneous data sources for users to not only perform "lookup", but also "learn" and "investigate". In terms of UI elements, relevant to our work is the Explorator system [7], that allows users to navigate through RDF data and run SPARQL queries by creation of facets and set operations. Our use of social guidance is also similar in nature to the social aspects of RExplorator [6], which extends Explorator with the ability to reuse results previously found by other users. Helix notably differs from the previously mentioned prior work [6, 7, 19, 28] in its support for navigation through heterogeneous data and its unique *online* linkage discovery capability.

6. CONCLUSION

In this paper, we described Helix, a system that allows knowledge workers and data scientists to *explore* a large number of data sources using a unified

intuitive user interface. Users can find portions of the data that are of interest to them using simple keyword search, navigate to other relevant portions of the data, and iteratively build customized views over one or more data sources. These features rely on highly scalable schema and linkage discovery performed as a pre-processing step, combined with online (and in part social) guidance on linkage and navigation. We demonstrated capabilities of our system through a number of usage scenarios.

We are currently working on extending Helix in a number of directions. On the user interface side, we are extending the social guidance feature through more complex query log analysis. Our goal is to predict a user's future steps through profiles based on similarity analysis of previous users' queries. We are also working on making user views accessible through a standard (RDF/SPARQL) API. This will allow non-expert users to build a custom and potentially complex knowledge graph, an alternative to the expensive, laborious task of building an enterprise-scale ontology for data analytics.

7. REFERENCES

[1] D. Agrawal, S. Das, and A. El Abbadi. Big Data and Cloud Computing: Current State and Future Opportunities. In *EDBT*, pages 530–533, 2011.

[2] M. A. Bornea, J. Dolby, A. Kementsietsidis, K. Srinivas, P. Dantressangle, O. Udrea, and B. Bhattacharjee. Building an Efficient RDF Store over a Relational Database. In *SIGMOD*, pages 121–132, 2013.

[3] A. Z. Broder. On The Resemblance and Containment of Documents. In *SEQUENCES*, pages 21–29. IEEE Computer Society, 1997.

[4] M. S. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *STOC*, pages 380–388, 2002.

[5] Peter Christen. A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *IEEE Trans. Knowl. Data Eng.*, 24(9):1537–1555, 2012.

[6] Marcelo Cohen and Daniel Schwabe. RExplorator - Supporting Reusable Explorations of Semantic Web Linked Data. In *ISWC Posters&Demos*, 2010.

[7] S.F.C. de Araújo and D. Schwabe. Explorator: A Tool for Exploring RDF Data through Direct Manipulation. In *LDOW2009*, 2009.

[8] X. Dong and A. Y. Halevy. Indexing Dataspaces. In *SIGMOD*, pages 43–54, 2007.

[9] S. Duan, A. Fokoue, O. Hassanzadeh, A. Kementsietsidis, K. Srinivas, and M. J. Ward. Instance-Based Matching of Large Ontologies Using Locality-Sensitive Hashing. In *ISWC*, pages 49–64, 2012.

[10] R. Goldman and J. Widom. Approximate dataguides. In *Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, pages 436–445, 1999.

[11] H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, and J. Goldberg-Kidon. Google Fusion Tables: Web-Centered Data Management and Collaboration. In *SIGMOD*, pages 1061–1066. 2010.

[12] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, pages 1–9, 2006.

[13] O. Hassanzadeh, F. Chiang, R. J. Miller, and H. C. Lee. Framework for Evaluating Clustering Algorithms in Duplicate Detection. *PVLDB*, 2(1):1282–1293, 2009.

[14] O. Hassanzadeh, S. Duan, A. Fokoue, A. Kementsietsidis, K. Srinivas, and M. J. Ward. Helix: Online Enterprise Data Analytics. In *WWW*, pages 225–228, 2011.

[15] O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller, and M. Wang. Semantic Link Discovery over Relational Data. In *Semantic Search over the Web*, pages 193–224. 2012.

[16] O. Hassanzadeh, K. Q. Pu, S. Hassas Yeganeh, R. J. Miller, M. Hernandez, L. Popa, and H. Ho. Discovering Linkage Points over Web Data. *PVLDB*, 6(6):444–456, 2013.

[17] S. Hassas Yeganeh, O. Hassanzadeh, and R. J. Miller. Linking Semistructured Data on the Web. In *WebDB*, 2011.

[18] V. Lopez, S. Kotoulas, M. L. Sbodio, M. Stephenson, A. Gkoulalas-Divanis, and P. M. Aonghusa. QuerioCity: A Linked Data Platform for Urban Information Management. In *ISWC*, pages 148–163, 2012.

[19] G. Marchionini. Exploratory Search: From Finding to Understanding. *CACM*, 49(4):41–46, 2006.

[20] S. Nestorov, J. D. Ullman, J. L. Wiener, and S. S. Chawathe. Representative Objects: Concise Representations of Semistructured, Hierarchial Data. In *ICDE*, pages 79–90, 1997.

[21] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4):334–350, 2001.

[22] M. Sayyadian, H. LeKhac, A. Doan, and L. Gravano. Efficient Keyword Search Across Heterogeneous Relational Databases. In *ICDE*, pages 346–355. 2007.

[23] Y. Sismanis, P. Brown, P. J. Haas, and B. Reinwald. GORDIAN: Efficient and Scalable Discovery of Composite Keys. In *VLDB*, pages 691–702, 2006.

[24] SPARQL Query Language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`.

[25] P. P. Talukdar, M. Jacob, M. S. Mehmood, K. Crammer, Z. G. Ives, F. Pereira, and S. Guha. Learning to Create Data-Integrating Queries. *PVLDB*, 1(1):785–796, 2008.

[26] T. Tran, H. Wang, S. Rudolph, and P. Cimiano. Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In *ICDE*, pages 405–416, 2009.

[27] M. A. Vaz Salles, J.-P. Dittrich, S. K. Karakashian, O. R. Girard, and L. Blunschi. iTrails: Pay-as-you-go Information Integration in Dataspaces. In *VLDB*, pages 663–674. 2007.

[28] R. W. White, S. M. Drucker, G. Marchionini, M. A. Hearst, and m. c. schraefel. Exploratory Search and HCI: Designing and Evaluating Interfaces to Support. Exploratory Search Interaction. In *CHI Extended Abstracts*, pages 2877–2880, 2007.

Report on the Seventh International Workshop on Business Intelligence for the Real Time Enterprise (BIRTE 2013)

Torben Bach Pedersen
Aalborg University
Denmark
tbp@cs.aau.dk

Malu Castellanos
HP Vertica
USA
malu.castellanos@hp.com

Umesh Dayal,
Hitachi Labs,
USA
umeshwar.dayal@hal.hitachi.com

1. INTRODUCTION

This paper reports on the 7th International Workshop on Business Intelligence for the Real Time Enterprise (BIRTE 2013), co-located with the VLDB 2013 conference. The BIRTE workshop series aims at providing a forum for presentation of the latest research results, new technology developments, and new applications in the areas of business intelligence and real time enterprises. Building on the success of the previous BIRTE workshops, co-located with the VLDB conferences in Seoul, Auckland, Lyon, Singapore, Seattle, and Istanbul, the seventh workshop in the series was held in Riva del Garda, Italy, on August 26, 2013.

Today, business analytics have to use new data sources and technologies in order for the business to be completely up-to-date. Traditional "in-house" data sources about transactions, sales, and finances still form the cornerstone of business analytics applications, but this is no longer enough. Instead, "Big Data" with high *velocity* such as tweets and other social network updates and sensor data from RFID, GPS, Bluetooth, etc. must be captured and analyzed instantly to understand the latest customer and market trends. Further, analyzing the past and even the present is no longer enough, so predictive analytics solutions are used to make decisions based on the expected future. These new applications and data sources mean that existing business intelligence methods and techniques must be revisited to provide better efficiency, scalability, expressiveness, and ease-of-use.

BIRTE 2013 featured an exciting technical program including two keynotes, an invited industrial talk, a panel, and a number of peer-reviewed papers from different countries in Europe, Africa, and Asia. Each submission received three reviews from the members of the distinguished program committee consisting of leading researchers in the field from academia and industry. From these submissions, two full research papers and one short position paper, along with two demo papers, were selected for presentation at the conference. Based on the feedback of the reviewers and the feedback at the workshop, the authors have made revised versions of their papers which will be published in a joint post-proceedings volume of BIRTE 2013 and 2014 in the Springer LNBIB series [1]. BIRTE 2013 was extremely well attended, with a peak audience of over 70 persons.

2. KEYNOTES

After the welcome by the BIRTE 2013 chairs, the program started with a keynote by Michael J. Carey from UC Irvine, entitled "AsterixDB: A New Platform for Real-Time Big Data BI". In this keynote, Prof. Carey explained the key ideas and principles behind the AsterixDB BDMS (Big Data Management System). AsterixDB has a number of features that sets it apart from other systems for managing Big Data. First, it has a unique flexible, semi-structured data model (Asterix Data Model) based on JSON. Second, is has a high-level declarative query language (AQL - Asterix Query Language) that can express a wide range of BI-like queries. Third, it has a highly scalable parallel runtime engine, Hyracks, that has been tested up to thousands of cores. Fourth, it supports new data intake very efficiently through its partitioned LSM-based data storage and indexing. Fifth, it has support for externally stored data (e.g., in HDFS) as well as natively managed data. Sixth, it features a rich set of primitive types, including spatial, temporal, and textual data types. Seventh, is has a range of secondary indexing options, including B+ tree, R tree, and inverted files. Eighth, is has support for fuzzy, spatial, and temporal queries as well as for parametric queries. Ninth, the notion of "datafeeds" supports continuous ingestion from relevant data sources. Finally, it has basic transactional capa-

bilities like those of a NoSQL data store. Asterix is a system where 'one size fits a bunch'.

The second keynote, by Prof. Johann-Christoph Freytag from Humboldt Universität zu Berlin was entitled "Query Adaptation and Privacy for Real-Time Business Intelligence" and aimed at taking a holistic view of the challenges and issues that relate to real-time business intelligence systems, by discussing both technical and non-technical aspects. First, the keynote introduced a number of real-world applications and used these to derive technical and non-technical requirements for real-time business intelligence. Based on these requirements and the experience of Prof. Freytag in co-developing the Stratosphere database management system with other Berlin research groups, the talk described techniques for query adaptation and histogram building in Stratosphere to support real-time business intelligence. The second part of the keynote discussed important aspects of privacy when dealing with personal data. It then outlined the necessary requirements for implementing real-time business intelligence systems to protect privacy, and discussed the trade-off between the level of privacy and the utility expected by those who perform real-time business analytics.

3. RESEARCH PAPERS

The next session featured two full research papers and a position paper. The paper "LinkViews: An Integration Framework for Relational and Stream Systems"' by Yannis Sotiropoulos and Damianos Chatziantoniou from Athens University of Economics and Business, addresses the current lack of a unified framework for querying (persistent) relational and stream data. Concretely, the authors proposed a view layer defined over standard relational systems to handle the mismatch between relational and stream systems. Here, database administrators define a special type of views (called LinkViews) which combine relational data and stream aggregates. The authors showed how this could achieve transparent integration of relations and streams and how queries could be optimized. Next, the paper "OLAP for Multidimensional Semantic Web Databases" by Adriana Matei, Kuo-Ming Chao, and Nick Godwin from Coventry University, proposed a new framework for doing OLAP over Semantic Web data. The framework has multiple layers including additional vocabulary, extended OLAP operators, and the SPARSQL query language, allowing the modeling of heterogeneous semantic web data, the unification of multidimensional structures, and enabling interoperability between different semantic web multidimensional databases. Finally, the paper "A Multiple Query Optimization Scheme for Change Point Detection on Stream Processing System" by Masahiro Oke and Hideyuki Kawashima from University of Tsukuba, showed how to apply multiple query optimization, well-known from relational database technology, to change point detection (CPD) queries. The authors propose a two-stage learning approach based on autoregressive models and divide CPD into four operators. To accelerate multiple CPD executions -needed for parameter tuning- they use multi-query optimization (MQO). The authors showed how MQO enables sharing a large part of the CPD processing, leading to significantly improved performance.

4. DEMOS

As a novel addition to the BIRTE program, two demo papers were presented. First, the demo paper "Big Scale Text Analytics and Smart Content Navigation" by Karsten Schmidt, Philipp Scholl, and Sebastian Bächle from SAP AG, and Georg Nold from Springer Science and Business Media, showed how to use the SAP Hana platform for flexible text analysis, ad-hoc calculations and data linkage. The goal is to enhance the experience of users navigating and exploring publications, and thus to support intelligent guided research in big text collections. Case data from the major scientific publisher Springer SBM was used. Second, the demo paper "Dynamic Generation of Adaptive Real-time Dashboards for Continuous Data Stream Processing" by Timo Michelsen, Marco Grawunder, Dennis Geesen, and H.-Jürgen Appelrath from University of Oldenburg presented a novel dashboard concept for visualizing the results from continuous stream queries, based on several individually configurable dashboard parts, each connected to a (user defined) continuous query, the results of which are received and visualized in real-time.

5. INDUSTRIAL INVITED TALK

Dr. Morten Middelfart from TARGIT gave an inspiring invited industrial talk on "The Inverted Data Warehouse based on TARGIT Xbone - How the biggest of data can be mined by the 'little guy'." The talk presented TARGIT's Xbone memory-based analytics server and defined the concept of an Inverted Data Warehouse (IDW), a DW storing query results rather than raw data. The concept and system were exemplified with a large-scale solution in which TARGIT Xbone and IDW were applied on Google search data with the aim of Search Engine Optimization (SEO).

6. PANEL

The workshop ended with a panel on "Real Time Analytics on Big Data" moderated by Meichun Hsu from HP Labs. The panel featured six distinguished panelists: Alejandro Buchmann from TU Darmstadt, Shel Finkelstein from SAP, Johann-Christoph Freytag from Humboldt University of Berlin, C. Mohan from IBM, Ippokratis Pandis from IBM, and Torben Bach Pedersen from Aalborg University. The panelists gave short presentations on their perspectives on the general topic and their responses to the four questions posed by the moderator: *what does real time analytics on big data really mean?, what are the compelling applications that motivated such capabilities? what is the status of the technology stack that delivers this capability and what are the gaps and challenges? Relative to the technology attributes often used to characterize big data such as extreme scale-out, NoSQL, and open source, and the emerging technologies such as SQL-on-Hadoop and in-memory stores, how do real time analytics relate?* After the presentations a lively (and somewhat controversial) debate ensued between the panelists and the highly active audience.

7. DISCUSSION AND OUTLOOK

We now summarize the discussions and contributions in the panel and the workshop overall, structured according to the four questions from the panel.

What does it mean? The first observation is that "real-time" is used with two different meanings: real-time as in streaming versus real-time as in agile business real-time, i.e., minutes/hours versus days. From a user perspetive, what really matters is to get current info/knowledge from data, i.e., get changes in the real world reflected in data asap. This means increasing data freshness demands and low query response times, but does not necessarily mean continuous queries/streams. It also means automatic notifications and responses to business events. For business real-time, it should be easy to ask new questions on new data, e.g., as supported by the paper on OLAP on Semantic Web data, enabling agile OLAP on new data sources. Another paper addressed stream query optimization. One paper combined the two meanings, business real-time and streaming, by auto-generating dashboards for streaming data. As for the Big Data buzz, the database industry has always worked on "bigger", the new value lies instead in using un- and semi-structured data. Finally, it was argued that "real-time" should not only mean the past and current, but also the future, i.e., tightly integrating forecasting and prediction with database and stream queries.

Compelling applications? The compelling applications discussed included CRM, brand sentiment, predictive maintenance, network optimization, security, fraud detection, text analytics and smart content navigation, the last two in an SAP paper. Major issues are discovering trends early and finding outliers. The analytics applications should be optimized for people, not machines, e.g., possibilities for user feedback are missing. A new type of applications concern cyber-physical systems producing huge amounts of data and events. One type of cyber-physical system is the emerging smart grid. Here, demand/supply flexibilities and forecasts must be tightly integrated and managed, as data is "born" in long-term forecasts, later re-forecasted, and finally measured and captured, before they are used as a basis for further planning and optimization.

Status of technology stack and relation to technology attributes? The last two questions are treated together. On the one hand, we see that several "new" data management technologies are emerging in this field. Examples include the AsterixDB system with its semi-structured data model, SAP Hana, based on main memory and compressed storage and offering integrated analysis and transactions in real-time in a single system, the Bubblestorm "data rendezvous" system which is self-organizing -correcting, -optimzing, and the TimeTravel system based on hierarchical models allowing efficient integrated querying of past, present, and future data. On the other hand, we see a reverse trend that "new" technologies such as column and compressed storage, vector processing, multi-core, and cache-awareness are now integrated into classical systems providing order of magnitude performance leaps, e.g., as exemplified by the DB2 Blu system. The technology stack is also aiming at integrating historical and streaming data as exemplified by the LinkView paper.

Finally, if we look at the topics listed in the Call for Papers that were not (in one way or another) discussed in the workshop, they were analytics as a service, cloud intelligence, collaborative real-time BI, crowdsourcing and crowd intelligence, and data quality and cleansing. The first two relate to running analytics as cloud-based services, which will be very relevant in the future, and the preferred option for many enterprises. However, this option is not wide-spread yet, which explains the lack of contributions for these topics. The next two topics are related to the role of people in real-time analytics, either a small-scale collaboration by analysts or a large-scale "collaboration" of many people in a crowd. Again, this is not yet done by most enterprises, but will surely become more used in the com-

ing years. Data quality and cleansing in the context of real-time analytics is a non-resolved topic, so papers on this topic will emerge.

In summary, we can conclude that business intelligence for the real-time enterprise is as relevant as ever, addressing in particular, the velocity aspect of Big Data and the new challenges imposed by this new trend. Thus, the outlook for BIRTE seems to be on the forward path with more editions planned for the future.

8. ACKNOWLEDGEMENTS

The BIRTE 2013 Chairs would like to thank all the authors of submitted papers for their interest in the workshop and the high quality of the submitted papers. We would also like to thank the distinguished PC members for their careful and dedicated work, both during the reviewing and the discussion phases. To the Organizing Committee of VLDB 2013, especially the General Chairs Themis Palpanas and Yannis Velegrakis, and the Workshop Chairs Tiziana Catarci, AnHai Doan, and Tova Milo, we would like to express our gratitude for their support to the BIRTE 2013 workshop. Finally, we would like to thank the BIRTE 2013 Proceedings Chair, Katja Hose and the web master Emmanouil Valsomatzis for their excellent job.

9. REFERENCES

[1] M. Castellanos, U. Dayal, K. Hose, T. B. Pedersen, and N. Tatbul (Eds.). Joint Proceedings of the 7th and 8th International Workshops on Business Intelligence for the Real Time Enterprise. Springer Lecture Notes in Business Information Processing, forthcoming.

Call for Papers
Special Issue on Visionary Ideas in Data Management
ACM SIGMOD Record

Guest Editor: Jun Yang
Editor-in-Chief: Yanlei Diao

This special issue of SIGMOD Record seeks papers describing visions of future systems, frameworks, algorithms, applications, and technology related to the management or use of data. The goal of this special issue is to promote the discussion and sharing of challenges and ideas that are not necessarily well-explored at the time of writing, but have potential for significantly expanding the possibilities and horizons of the field of databases and data management. The submissions will be evaluated on their originality, significance, potential impact, and interest to the community, with less emphasis on the current level of maturity, technical depth, and evaluation.

Important Dates

Submission deadline: **March 15, 2015**
Publication of the special issue: **June 30, 2015**

Submission Guidelines

Submissions should be 6 pages in length, formatted according to the current SIGMOD Record guidelines (two-column, 10pt Times New Roman font). Papers should be submitted electronically in PDF format through the SIGMOD RECord Electronic Submission System (RECESS) at

http://sigmod.hosting.acm.org/record/index.php

Please select "[Special Issue] Vision Articles" as the column you are submitting to.

For detailed preparation and submission instructions, see below (Formatting Guidelines).

http://www.sigmod.org/publications/sigmod-record/authors