SIGMOD Officers, Committees, and Awardees

Chair

Yannis Ioannidis
University of Athens
Department of Informatics
Panepistimioupolis, Informatics Bldg
157 84 Ilissia, Athens
HELLAS
+30 210 727 5224
<yannis AT di.uoa.gr>

Vice-Chair

Christian S. Jensen
Department of Computer Science
Aarhus University
Åbogade 34
DK-8200 Århus N
DENMARK
+45 99 40 89 00
<csj AT cs.aau.dk >

Secretary/Treasurer

Alexandros Labrinidis
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260-9161
PA 15260-9161
USA
+1 412 624 8843
<labrinid AT cs.pitt.edu>

SIGMOD Executive Committee:

Sihem Amer-Yahia, Curtis Dyreson, Christian S. Jensen, Yannis Ioannidis, Alexandros Labrinidis, Maurizio Lenzerini, Ioana Manolescu, Lisa Singh, Raghu Ramakrishnan, and Jeffrey Xu Yu.

Advisory Board:

Raghu Ramakrishnan (Chair), Yahoo! Research, <First8CharsOfLastName AT yahoo-inc.com>,
Amr El Abbadi, Serge Abiteboul, Rakesh Agrawal, Anastasia Ailamaki, Ricardo Baeza-Yates, Phil Bernstein, Elisa Bertino, Mike Carey, Surajit Chaudhuri, Christos Faloutsos, Alon Halevy, Joe Hellerstein, Masaru Kitsuregawa, Donald Kossmann, Renée Miller, C. Mohan, Beng-Chin Ooi, Meral Ozsoyoglu, Sunita Sarawagi, Min Wang, and Gerhard Weikum.

Information Director, SIGMOD DiSC and SIGMOD Anthology Editor:

Curtis Dyreson, Washington State University, <cdyreson AT eecs.wsu.edu>

Associate Information Directors:

Denilson Barbosa, Ugur Cetintemel, Manfred Jeusfeld, Georgia Koutrika, Alexandros Labrinidis, Michael Ley, Wim Martens, Rachel Pottinger, Altigran Soares da Silva, and Jun Yang.

SIGMOD Record Editor:

Ioana Manolescu, INRIA Saclay, <ioana.manolescu AT inria.fr>

SIGMOD Record Associate Editors:

Magdalena Balazinska, Denilson Barbosa, Pablo Barceló, Vanessa Braganholo, Chee Yong Chan, Anish Das Sarma, Ugur Çetintemel, Brian Cooper, Cesar Galindo-Legaria, Glenn Paulley, Alkis Simitsis, Nesime Tatbul and Marianne Winslett.

SIGMOD Conference Coordinator:

Sihem Amer-Yahia, CNRS and LIG, France, <sihemameryahia AT acm.org>

PODS Executive Committee: Rick Hull (chair), <hull AT research.ibm.com>, Michael Benedikt, Wenfei Fan, Maurizio Lenzerini, Jan Paradaens and Thomas Schwentick.

Sister Society Liaisons:

Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment).

Awards Committee:

Rakesh Agrawal (Chair), Microsoft Research, <rakesh.agrawal AT microsoft.com>, Elisa Bertino, Peter Buneman, Umesh Dayal and Masaru Kitsuregawa.

Jim Gray Doctoral Dissertation Award Committee:

Johannes Gehrke (Co-chair), Cornell Univ.; Beng Chin Ooi (Co-chair), National Univ. of Singapore, Alfons Kemper, Hank Korth, Alberto Laender, Boon Thau Loo, Timos Sellis, and Kyu-Young Whang.

[Last updated : June 30th, 2012]

Alfons Kemper, Hank Korth, Alberto Laender, Boon Thau Loo, Timos Sellis, and Kyu-Young Whang.

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. Until 2003, this award was known as the "SIGMOD Innovations Award." In 2004, SIGMOD, with the unanimous approval of ACM Council, decided to rename the award to honor 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)

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)

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, University of Toronto; Yanlei Diao, University of California at Berkeley.
- **2007** *Winner*: Boon Thau Loo, University of California at Berkeley. *Honorable Mentions*: Xifeng Yan, University of Indiana at Urbana Champaign; Martin Theobald, Saarland University
- **2008** *Winner*: Ariel Fuxman, University of Toronto. *Honorable Mentions*: Cong Yu, University of Michigan; Nilesh Dalvi, University of Washington.
- **2009** *Winner*: Daniel Abadi, MIT. *Honorable Mentions*: Bee-Chung Chen, University of Wisconsin at Madison; Ashwin Machanavajjhala, Cornell University.
- **2010** *Winner:* Christopher Ré, University of Washington. *Honorable Mentions*: Soumyadeb Mitra, University of Illinois, Urbana-Champaign; Fabian Suchanek, Max-Planck Institute for Informatics.
- **2011** *Winner*: Stratos Idreos, Centrum Wiskunde & Informatica. *Honorable Mentions*: Todd Green, University of Pennsylvania; Karl Schnaitter, University of California in Santa Cruz.
- **2012** *Winner*: Ryan Johnson, Carnegie Mellon University. *Honorable Mention*: Bogdan Alexe, University of California in Santa Cruz.

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

[Last updated : June 30th, 2012]

Editor's Notes

Welcome to the June 2012 issue of the ACM SIGMOD Record!

The issue opens with the Database Principles column, where Mikolaj Bojanczyk explores algorithms based on algebras (and in particular, on monoids) for solving word and tree problems, such as evaluating binary queries on words and forests. The paper by Peter Wood surveys query languages for graph databases, a topic on which attention is renewed due to the popularity of important applications such as Semantic Web graphs of RDF data, and social network graph analysis. Query languages are analyzed from the viewpoint of features (expressive power) and then under the angle of their associated algorithmic complexity.

The survey by Fereira, Gonçalves and Laender focuses on an important problem in scholarly and bibliographic data management systems: automatic author name disambiguation. This problem is a well-known particular case of data cleaning, where errors can be due to erroneous data entry or to natural changes in the names and affiliation an author has along her or his career. In other contexts, two different authors may actually have the exact same name, complicating the task of setting their publications apart. The survey formalizes the problem and outlines two main computational techniques used to solve it: author grouping attempts to group the references of the same author according to some similarity, while author assignment aims at directly assigning each reference to the right author. Finally, the survey places a wide selection of relevant work in the classification space thus defined.

The issue includes two Distinguished Profiles in Databases. Tamer Ozsu, recipient of the SIGMOD Contribution Award of 2006 and an ACM Fellow, muses on the spirit of object-oriented database hovering above current database modeling tools, even as pure object-oriented databases have by now disappeared. Tamer is also known to have voiced strong opinions in the conference versus journal discussions, and in this interview he explains how the database community has overplayed the role of conferences and should join the majority of the scientific community in giving more importance to conferences. If you have hated 8-pages major conference submissions as much as I did, it is time to learn Tamer's arguments in favor of it! Tamer also discusses the PVLDB model, the Springer database encyclopedia, left-wing politics, his 600-strong fountain pen collection, and more!

Our second Profile features Erich Neuhold, now a professor at the University of Vienna and a director of the Fraunhofer Institute for Integrated Publication and Information Systems in Darmstadt. Just like Tamer, Erich has been involved over time in object-oriented and XML databases, and he discusses his own lessons learned in this area. Erich also comments on his experience managing a large industrial and then a larger University department, and draws an interesting comparison between research in the US versus Europe. Erich shares many insights into the historical evolution of German and other European research institutes, in particular from the perspective of their funding. The interview also recalls the days when 64 K seemed more memory than a program would ever need, and about the innovative idea of connecting ten such computers into a distributed database system!

The issue closes with two calls for contributions. The VLDB Journal editors-in-chief want it to be known that survey submissions are still very much welcome to the Journal! Closing the issue is the call for papers for ICDT 2013, to be held in Genoa, Italy, early next year.

Changes to the editorial board Last but not least, I am pleased to announce the presence of new members in the SIGMOD Record Editorial Board:

- Anish das Sarma (Google) joined us to help Brian Cooper edit submissions to the Reports column;
- Nesime Tatbul (ETH Zurich) is the new survey co-editor, helping Cesar Galindo-Legaria;
- Alkis Simitsis (HP Labs) joins Ugur Cetintemel as a co-editor of the Research Centers column, with the plan that Alkis will take over from Ugur in a short time.

Welcome to the new editors and may they enjoy a happy tenure!

Your contributions to the Record are welcome via the RECESS submission site (http://db.cs.pitt.edu/recess). Prior to submitting, be sure to peruse the Editorial Policy on the SIGMOD Record's Web site (http://www.sigmod.org/publications/sigmod-record/sigmod-record-editorial-policy).

Ioana Manolescu

June 2012

Past SIGMOD Record Editors:

Harrison R. Morse (1969)
Daniel O'Connell (1971 – 1973)
Randall Rustin (1975)
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)

Algorithms for regular languages that use algebra

Mikołaj Bojańczyk*
University of Warsaw

ABSTRACT

This paper argues that an algebraic approach to regular languages, such as using monoids, can yield efficient algorithms on strings and trees.

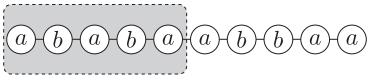
1. INTRODUCTION

A practically minded reader might have doubts about using monoids and other algebras for regular languages, instead of automata. The goal of this paper is to show that algebra is actually quite straightforward and can be practically useful in algorithms for words and trees.

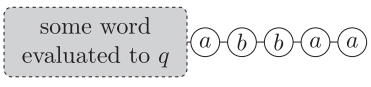
2. MONOIDS

A monoid is a method of recognizing regular languages, which is an alternative to automata and regular expressions. We illustrate the differences between recognizing a language with a deterministic finite automaton and a monoid.

In a deterministic automaton, a state $q \in Q$ represents a prefix of an input word:

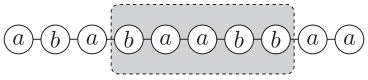


The state captures all the relevant information that the automaton needs to know about the prefix. This can be stated as the following compositionality principle: swapping a prefix with another prefix that gives the same state does not affect the run of an automaton on the remaining part of the input. Therefore, a prefix can be abstracted away as just its state:



*Supported by FET-Open grant agreement FOX, number FP7-ICT-233599. I would like to thank Jakub Łącki and an anonymous referee for comments that improved this paper.

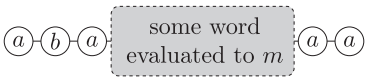
In the monoid approach, instead of an automaton, we have a function α, which maps every word w to an element of a set M, called the monoid. The idea is that the function α can be applied not just to prefixes of the input, but also to infixes:



The values assigned by the function α to an infix should capture all the relevant information about the infix. This can be stated as the following compositionality principle: swapping an infix with another infix of the same value does not affect the value of the whole input. In other words,

$$\alpha(v) = \alpha(v') \quad \text{implies} \quad \alpha(w_1 v w_2) = \alpha(w_1 v' w_2)$$

holds for all words $v, v', w_1, w_2 \in A^*$. A function that is compositional in the above sense is called a *monoid morphism*. In a monoid morphism α, an infix of the input can be abstracted away by its value under the monoid morphism:



We say that a monoid morphism $\alpha : A^* \to M$ recognizes a language $L \subseteq A^*$ if there is some set $F \subseteq M$ of accepting elements such that a word belongs to L if and only if it is mapped to F. A monoid morphism can recognize several languages, depending on the choice of F.

Examples of monoid morphisms and languages that they recognize include:

- The function $\{a, b\}^* \to \{0, 1\}$ which maps a word to 1 if and only if it contains some a. If we choose the accepting set F as $\{0\}$, then the morphism recognizes the language b^*.
- The function which maps a word to its length. The monoid used is not finite. This morphism

recognizes, e.g. the set of words of prime length. We are not interested in this kind of monoid morphism, because we only care about finite monoids.

- For $n \in \mathbb{N}$, the function

$$\alpha_n : \{a,b\}^* \to \{a,b\}^{\leq n}$$

which maps words of length $< n$ to themselves, and longer words to their prefix of length n. This morphism recognizes the language "the n-th letter is a". The set $\{a,b\}^{\leq n}$ is exponential in n; it is not difficult to see that this language cannot be recognized using a smaller set. A deterministic automaton for this language needs only $n+2$ states, since it corresponds to the function

$$\beta_n : \{a,b\}^* \to \{a,b\} \cup \{0,\ldots,n-1\}$$

which maps words of length $< n$ to their length, and longer words to their n-th letter. The function β_n is not a monoid morphism, because

$$\alpha(a^{n-1}) = \alpha(b^{n-1}) \quad \text{but} \quad \alpha(aa^{n-1}) = \alpha(ab^{n-1})$$

A corollary of compositionality is that for every words w_1, w_2, the value $\alpha(w_1 w_2)$ depends only on the values of $\alpha(w_1)$ and $\alpha(w_2)$. The operation

$$(\alpha(w_1), \alpha(w_2)) \in M^2 \qquad \mapsto \qquad \alpha(w_1 w_2) \in M$$

is called the *monoid operation* in M. If the monoid morphism is surjective, the operation is defined on all pairs in M^2. It is not difficult to see that because of compositionality, the monoid operation must be associative, which means that the result of the monoid operation for a sequence $m_1, \ldots, m_k$ does not depend on its bracketing. Finally, the image of the empty word is a neutral element for the monoid operation.

A monoid morphism is uniquely represented by: the images of single letters and the empty word, and the multiplication table for the monoid operation.

From automata to monoids and back. Monoid morphisms are just another way of talking about regular languages.

THEOREM 1. *A language $L \subseteq A^*$ is regular if and only if it is recognized by a morphism into a finite monoid.*

PROOF. Suppose that L is recognized by a monoid morphism $\alpha : A^* \to M$. Then one constructs an automaton, with states M, which maps every infix of the input to its value under α.

A bit more effort is required to go from an automaton to a monoid. The construction even works for nondeterministic automata. Consider a nondeterministic automaton with states Q. Consider a function which maps a word $w \in A^*$ to the set

$$\{(p,q) \in Q^2 : \text{ some run over } w \text{ goes from } p \text{ to } q\}.$$

It is not difficult to see that this mapping is compositional in the monoid sense. □

As seen in the above proof, the conversion from a monoid to a deterministic automaton, is linear. In the other direction, sometimes the exponential explosion from the theorem's proof cannot be avoided, as witnessed by the example "the n-th letter is a". Languages, for which the monoid has approximately the same size as a nondeterministic automaton include "the word contains at least n letters a".

The exponential blowup from automata to monoids is a problem for some algorithms; but it will not be a problem for the algorithms presented in this paper.

Expository Note. Typically, monoids are introduced as follows. One defines a monoid as an abstract algebraic structure, which is a carrier M equipped with an associative binary operation and a neutral element. Then one observes that the set of all words is a monoid. Finally, one defines a monoid morphism $\alpha : A^* \to M$ to be any function which preserves the structure of a monoid, namely the monoid operation and the monoid identity.

In this paper, a monoid is presented in the opposite order: we have seen that any surjective mapping from words to a set M which is compositional in the appropriate sense uniquely determines a structure of a monoid in its image M. The reason for this choice is that sometimes it is easier to think of a compositional mapping than an abstract algebraic structure.

3. ALGORITHMS FOR WORDS

As explained in the previous section, a monoid morphism assigns values to all infixes of an input word, and not just the prefixes. This means that a monoid morphism is a more flexible structure, and it is better suited to some algorithms. We illustrate this on several examples.

3.1 Incremental updates

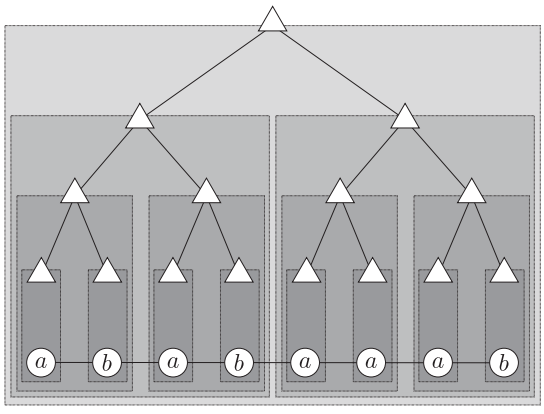
We begin with a very straightforward algorithm, which introduces a data structure that is heavily used in the rest of the paper. The data structure is a hierarchical decomposition of the input, decorated by values of a monoid morphism.

Here is the problem we want to solve. Let $L \subseteq A^*$ be a regular language of words. We begin with some initial word in $a_1 \cdots a_n \in A^*$. A user produces a sequence of edits of the form "change the label of letter $i \in \{1, \ldots, n\}$ to a". We want an algorithm which can tell, after a sequence of edits, if the current state of the word belongs to L. This problem can be solved with linear preprocessing and logarithmic cost per edit.

THEOREM 2. *Let $L \subseteq A^*$ be a regular language. There is an algorithm which:*

1. *inputs an initial word and builds a data structure in linear time;*
2. *receives a sequence of edits and updates the data structure in logarithmic time for each edit;*
3. *can tell in constant time, using the data structure, if the word after the edits belongs to L.*

PROOF. The algorithm uses a straightforward tree decomposition approach. Let $a_1 \cdots a_n \in A^*$ be the initial word. Divide the set of all positions in the word into two infixes, of approximately the same lengths, and then do this division recursively for the infixes. As a result, we get a tree decomposition of the initial word, as depicted in the following picture, where the circles denote positions in the word, and the triangles denote nodes in the tree (and nodes correspond to infixes):



.

In general, if the length of the initial word is not a power of two, then not all leaves have the same depth, but the maximal depth is still logarithmic.The tree has at most $2n$ nodes and can be computed in linear time (of course, we store the infixes by keeping their first and last positions).

Suppose that α is a monoid morphism which recognizes the language L. Such a monoid morphism exists by Theorem 1. For a node x of the tree, let us denote by w_x the infix of the word that corresponds to x. Because a node x with children y and z satisfies

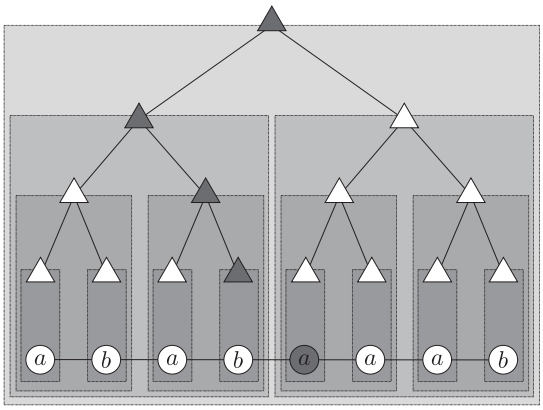
$$\alpha(w_x) = \alpha(w_y) \cdot \alpha(w_z), \tag{1}$$

we can do leaf-to-root pass through the tree t and compute in time linear in $|w|$ the values

$$\{\alpha(w_x)\}_{x \in nodes(t)}.$$

The tree t together with the values (1) is our data structure. We have just argued that it can be computed in linear time.

Consider an edit operation, which changes the label of letter $i \in \{1, \ldots, n\}$. The infixes in the data structure that contain i form a root-to-leaf path in the tree t, which is illustrated by a darker shade of gray in the following picture:



This path has logarithmic length. We only need to change the labels for these infixes, in a leaf-to-root pass. We can do this in logarithmic time using (1).

Finally, if we want to know if the current word belongs to the language L, we just need to look at the root of the tree, and see if the monoid element stored there belongs to the accepting set. □

When describing the running time of the algorithm in Theorem 2, we assumed that the language L was fixed. Suppose now that the language L is also given on the input, and represented by a nondeterministic automaton with states Q. What is the running time of the algorithm, in terms of both the input word $a_1 \cdots a_n$ and the state space Q? We claim that the data structure is built in time

$$poly(Q) \cdot O(n),$$

and each update operation is processed in time

$$poly(Q) \cdot \log n.$$

This is despite the fact that the algorithm uses monoids, and that monoids can be exponentially

larger than automata. The reason is that the algorithm does not need to compute the whole monoid. It only needs to store elements of the monoid in the tree nodes; and the space to store a monoid element is logarithmic in the size of the monoid, and therefore polynomial in Q. The algorithm also needs to consult the multiplication table of the monoid, but this multiplication table can be generated on-the-fly, in time polynomial in Q.

3.2 Evaluation of binary queries

In this section, we adapt the algorithm from Theorem 2 to evaluate binary queries on words.

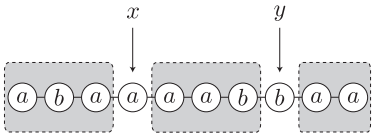
Binary queries. A binary query on words over alphabet A is a function φ, which maps every word $w \in A^*$ to a set of pairs of nodes in w. Typical queries include:

- Select pairs $x \leq y$ such that there is at least one a in the interval $\{x, \ldots, y\}$.
- Select pairs $x \leq y$ such that there an even number of a's in the interval $\{x, \ldots, y\}$.

In this paper, we are interested in regular binary queries, which are the query equivalent of regular languages. Just like for regular languages, there are multiple ways of defining regular queries. We use a definition based on monoids.

Binary query recognized by a monoid morphism. Let x, y be two distinct positions in a word w. There are two possible scenarios:

1. x is before y. In this case, the positions of the word can be partitioned into: x, y, and three infixes as in the picture below:



2. The second scenario is when x is after y. A decomposition similar to the one in the first scenario exists.

Suppose that $\alpha : A^* \to M$ is a monoid morphism. We define the α-type of a pair (x, y) of distinct positions in a word $w \in A^*$ to be the following information: which scenario holds, what are the labels of x and y, what are the values under α of the three infixes in the scenario. The α-type belongs to

$$\{<, >\} \times A^2 \times M^3$$

A monoid morphism is said to recognize a binary query if one can choose a subset F of accepting α-types, such that for every word and every pair of positions, the pair of positions satisfies the query if and only if its α-type belongs to F.

We say that a binary query is regular if it is recognized by some monoid morphism. The examples at the beginning of Section 3.2 are regular. Also, the class of regular binary queries coincides with the class of binary queries that can be defined by formulas of monadic second-order logic with two free first-order variables.

THEOREM 3. *Let φ be a regular binary query. There is an algorithm which:*

1. *inputs a word $w \in A^*$ and builds a data structure in linear time;*
2. *using the data structure, answers in logarithmic time questions of the form: is the pair (i, j) selected by φ?*

PROOF. Suppose that the binary query is recognized by a monoid morphism $\alpha : A^* \to M$. We use the same data structure as in Theorem 2.

The key observation is the following one: every infix v of the input word can be decomposed as a concatenation

$$v = v_1 \cdots v_k$$

of at most logarithmically many infixes $v_1, \ldots, v_k$ that correspond to nodes in the data structure. This decomposition can be computed in logarithmic time, given the first and last position in the infix v. By this observation, we can use the data structure to compute the α-type of any pair of positions in logarithmic time. □

4. SIMON FACTORIZATION

The algorithms we have seen so far used a tree-like decomposition of the input word, of logarithmic depth. What if we could have a constant depth decomposition, with the depth only depending on the monoid morphism, and not on the input word?

In this section, we provide such a constant depth decomposition. The underlying result is called the *Simon Factorization Forest Theorem*. This section is where monoids and their theory start to do some real work.

Simon factorization trees. Let $\alpha : A^* \to M$ be a monoid morphism. Let w be a word with n positions. A Simon α-factorization tree is similar to the tree from Theorem 2 in the following ways:

- Every node of x of the tree corresponds to an infix w_x of the word w. The leaves correspond to single letters; the root corresponds to the whole word w.
- If a node x has children $x_1, \ldots, x_n$ then
$$w_x = w_{x_1} \cdots w_{x_n}.$$
- Every node x stores the value $\alpha(w_x) \in M$.

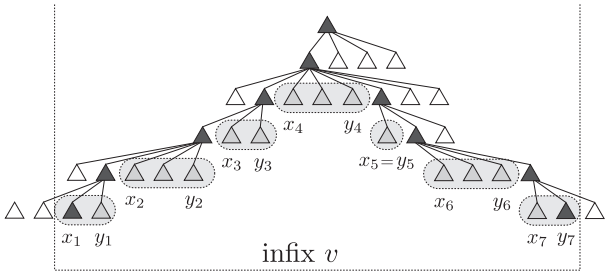
The difference is that in a Simon factorization tree, a node can have more than two children. Having an unbounded number of children is necessary if we want to have trees of constant depth for words of unbounded length. In a Simon factorization tree, there is a restriction for nodes x with three or more children:

- If x has at least three children $x_1, \ldots, x_n$, then
$$\alpha(w_{x_1}), \ldots, \alpha(w_{x_n})$$
are the same element, call it $m \in M$. Furthermore, m is idempotent, which means that $m \cdot m = m$. It follows that
$$\alpha(w_x) = \alpha(w_{x_1}) \cdots \alpha(w_{x_n}) = m \cdots m = m.$$

Consider a Simon α-factorization tree as in the definition above. Let x and y be siblings such that y is to the right of x. Let
$$x = z_1, \ldots, z_k = y$$
be all of the siblings between x and y. Define
$$w_{x \ldots y} = w_{z_1} \cdots w_{z_k}.$$
Observe that the monoid element $\alpha(w_{x \ldots y})$ can be computed in constant time; as a function of x and y. Indeed, when $k = 1, 2$, then $\alpha(w_{x \ldots y})$ is $\alpha(w_x)$ and $\alpha(w_x) \cdot \alpha(w_y)$, respectively. The more interesting case is when $k \geq 3$. In this case, the parent of x and y has at least three children. Let m be the (idempotent) monoid element stored in the parent. By the definition of Simon factorization trees,
$$\alpha(w_{x \ldots y}) = \alpha(w_{z_1}) \cdots \alpha(w_{z_k}) = m.$$

LEMMA 1. *For every infix v one can compute a sequence of sibling pairs*
$$(x_1, y_1), \ldots, (x_m, y_m)$$
such that
$$v = w_{x_1 \ldots y_1} \cdot w_{x_2 \ldots y_2} \cdot \ldots \cdot w_{x_m \ldots y_m}$$
The number of sibling pairs and the time to compute them is proportional to the depth of the tree.

PROOF BY PICTURE.



□

COROLLARY 1. *The value under α of an infix can be computed in time linear in the depth of the tree*[1].

PROOF. We use the decomposition of the infix from Lemma 1. For each sibling interval, its value under α can be computed in constant time: either it corresponds to at most two nodes, or its value is the same as the value in the parent of the interval. □

Constant depth. The key result about Simon factorization trees is that they can be built so that their height depends only on the monoid, and not the length input word:

THEOREM 4. *Let $\alpha : A^* \to M$ be a monoid morphism. Every word $w \in A^*$ has a Simon α-factorization tree of depth at most $3|M|$, which can be computed in time $poly(M) \cdot |w|$.*

The proof of this theorem, although not difficult, relies on results about monoids that cannot be easily recovered by treating monoids as a decoration on automata.

From Corollary 1 and Theorem 4 it follows that the algorithm from Theorem 3 can be improved from logarithmic time to constant time query evaluation:

THEOREM 5. *Let φ be a regular binary query. There is an algorithm which:*

1. *inputs a word $w \in A^*$ and builds a data structure in linear time;*
2. *using the data structure, answers in constant time questions of the form: is the pair (i, j) selected by φ?*

PROOF. The same proof as for Theorem 3, but use Corollary 1 to compute the values of infixes. □

[1]By adding accelerating pointers, this can be improved to time logarithmic in the depth of the tree, see [4]

Incremental updates. We have just shown how to use Theorem 4 to improve the algorithm for binary query evaluation. What about the algorithm for incremental updates? Could Simon factorization trees be used to get an algorithm that processes edits in constant time? Unfortunately [6] shows that some languages require at least

$$\frac{\log n}{\log \log n}$$

operations per edit.

4.1 References

A survey on the Simon factorization theorem, and some other applications for algorithms, can be found in [3]. The original version of Theorem 4 is from [9]. The $3|M|$ bound on the depth of the factorization tree, which is optimal, comes from [7]. A version of the theorem where the factorization tree is constructed by a deterministic left-to-right automaton is shown in [5]. The case when the monoid is obtained from a nondeterministic automaton is studied in [4].

5. REGULAR TREE LANGUAGES

We now move from regular languages of words to regular languages of trees. We used node-labelled, unranked (which means that there is no bound on the number of children), sibling-ordered trees, which are a common model for XML documents.

Nondeterministic tree automata. A *nondeterministic tree automaton* is given by the following ingredients:

1. An *input alphabet* A;
2. A set of states Q, with a distinguished accepting subset $F \subseteq Q$ of *root accepting* states.
3. A finite set δ of *transitions*. Each transition is a triple (q, a, L) where $q \in Q$, $a \in A$, and $L \subseteq Q^*$ is a regular language.

Of course, when representing such an automaton, one needs to choose some representation for the regular word languages in the transitions, e.g. by using regular expressions or maybe nondeterministic word automata. This choice of representation influences the complexity of algorithms that deal with the automata.

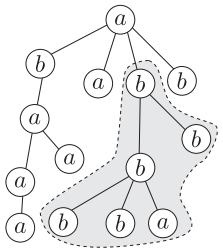
Consider an input tree t. An *run* of such an automaton is a labeling $\rho : nodes(t) \to Q$ such that for every node x with children $x_1, \ldots, x_n$ written from left to right, there exists a transition $(q, a, L) \in \delta$ such that the run maps x to q, the label of x is a, and the language L contains the word $\rho(x_1) \cdots \rho(x_n)$. In the special case when x is a leaf, the language L should contain the empty word. This special case explains why initial states are not needed in the definition of the automaton.

A tree is *accepted* if there is a run where the root is mapped to a root accepting state. The language *recognized* by an automaton is the set of accepted trees. A tree language is called *regular* if it is recognized by some automaton.

Just like for regular word languages, the class of regular tree languages is very robust and can be described in many other equivalent ways, including deterministic/alternating automata, Myhill-Nerode equivalence and monadic second-order logic[2].

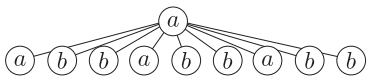
6. FOREST ALGEBRA

Just as in the case of words, an automaton can be seen as a device which assigns a value to parts of its input. In the case of nondeterministic tree automaton, the parts which get a value are subtrees, such as in the following picture:



Since we are using a nondeterministic automaton model, one should think of a subtree as being mapped to a set of states.

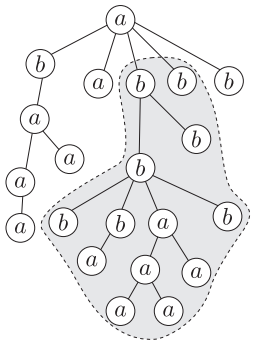
Forests. Subtrees are not general enough for the purposes of some of the algorithms we use. They suffer from the same problems as prefixes (as opposed to infixes) for words, together with some new problems. For instance, if you are only allowed to use subtrees, then how are you going to hierarchically decompose a tree with one root and many children, such as the one below?



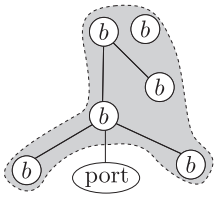
That is why our basic object of interest will not be a subtree, but an (ordered) forest, which is an ordered

[2]One of the choices of definition that *does not* lead to regular tree languages is the very natural model of deterministic root-to-leaves automata. Deterministic leaves-to-root automata, on the other hand, are equivalent to nondeterministic ones, and therefore to regular tree languages.

sequence of trees. In other words, a forest is a tree with multiple (ordered) roots. Inside a bigger forest we can find a smaller forest, by distinguishing a set of nodes called *forest zone*. A forest zone is a set of nodes in a forest which is closed under descendants, and such that the roots of the zone (which means the least nodes with respect to the ancestor relation) form a sequence of consecutive siblings. Here is a picture of a forest zone inside a tree

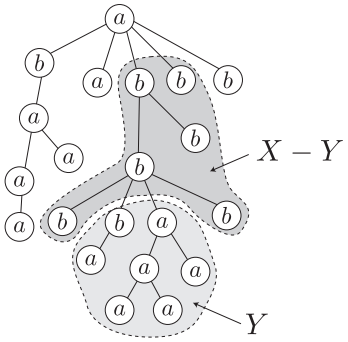


Contexts. Forests alone are also not general enough for the purposes of our algorithms. Recall that in Theorem 2, we used a data structure where each infix was split into two infixes, and so on recursively. This will not work for forests — for instance a tree cannot be split into two forests in any way. That is why we use a second kind of object, which is called a *context*. Formally, a context is a forest with one distinguished leaf, which is called a *port*, as in the following picture:

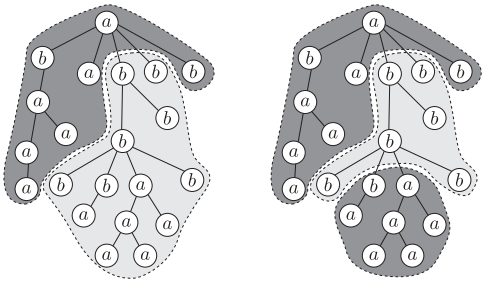


The idea is that the port can be filled by any forest. After the port in the picture above is filled by a forest with n roots, then parent of the port will have $n + 2$ children (because there are already two siblings of the port).

A context can be seen as a part of a forest, by distinguishing a set of nodes called *context zone*. A context zone inside a forest is defined as a difference of two forest zones $X - Y$. Here is a picture:



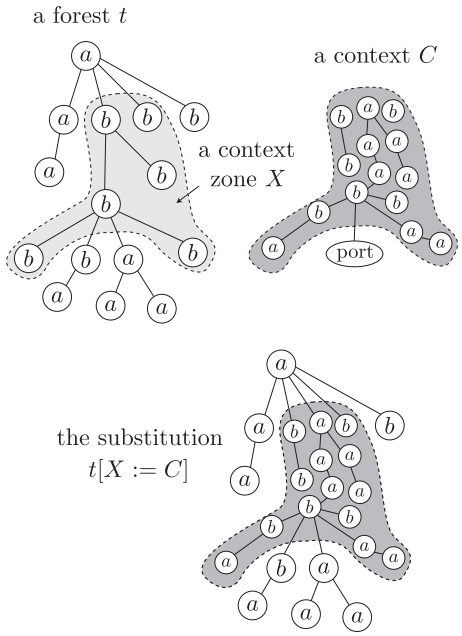
Partitioning zones. We claim that forest and context zones are general enough to do decompositions. As a first example, consider the complement of a (forest or context) zone. Although the complement is not necessarily a zone itself, it can be partitioned into at most two zones. The following pictures illustrates a forest (respectively, context) zone in light gray, and its complement in darker gray.



Generally speaking, since zones are sets of nodes, we can do the boolean operations on them: union, intersection and complementation. Although the result of a boolean combination of zones is not necessarily a zone itself, it can be partitioned into a small number of zones, as the following lemma shows.

LEMMA 2. *A boolean combination of n zones can be partitioned into $O(n)$ zones.*

Substitution. Suppose that we have a forest t and a context C. If we distinguish a context zone X inside t, then we can replace the contents of that context zone by C, with the result being called $t[X := C]$. This process is illustrated below:



In general, there are four kinds of substitution: the enclosing object can be a context or a forest, and the substituted object can be a context or forest.

Forest algebra morphism. We now introduce forest algebra. Forest algebra is for forests and contexts, as monoids are for infixes. Like we did for monoids, we focus on the morphisms. A forest algebra morphism over the alphabet A consists of two sets (H, V) and functions:

1. a function α_H from forests over A to H;
2. a function α_V from contexts over A to V;

which are compositional in the following sense. Let t be a forest, X a context zone inside t, and let C and C' be two contexts. Then

$$\alpha_V(C) = \alpha_V(C') \quad \text{implies}$$
$$\alpha_H(t[X := C]) = \alpha_H(t[X := C']).$$

Likewise for the other three kinds of substitutions. In other words, all we need to know about a zone (forest or context) is its value under α_H or α_V.

A forest algebra morphism can be used to recognize a set of forests. This is done by distinguishing an accepting subset $F \subseteq H$; the recognized language is then the set of forests that are mapped to F by the morphism. If you want to recognize a tree language, you should make sure that only trees are mapped to elements of F.

The following theorem shows that forest algebras, as a recognizing device for tree languages, are equivalent to automata.

THEOREM 6. *A tree language is recognized by a nondeterministic tree automaton if and only if it is recognized by a forest algebra morphism.*

As in the word case, there is a (singly) exponential blowup when going from a nondeterministic automaton to a forest algebra morphism.

6.1 References

For a survey on regular tree languages with a database angle, see [8]. A discussion of forest algebra and other algebras for trees can be found in [2].

7. ALGORITHMS FOR TREES

In this section, we show how forest algebra can be used to generalize from words to trees the algorithms that we have seen in Section 3.

7.1 Hierarchical decompositions of forests

The data structure that we used in Section 3 was a hierarchical decomposition, where the word was split into halves, quarters, and so on. We now show that a similar decomposition is possible for forests.

LEMMA 3. *Every (forest or context) zone X can be partitioned into at most four (forest or context) zones such that each of the parts has at most $2/3 \cdot |X|$ nodes.*

PROOF. Let n be the size of X.

We say that a node $x \in X$ is small if at most one third of the nodes of X are descendants of x. Suppose that not all nodes in X are small (the degenerate case when all nodes are small is treated the same way). There must be a node $x \in X$ which is not small, but which has only small children, call them $x_1, \ldots, x_k$. For each $i \in \{1, \ldots, k\}$, define D_i to be the nodes in X that are (not necessarily strict) descendants of one of the nodes $x_1, \ldots, x_i$. The set D_1 has at most $n/3$ nodes and the set of D_k has at least $n/3 - 1$ nodes, because otherwise the parent x would be small. Take i to be the first index such that D_i has at least $n/3$ nodes. Then

$$\frac{n}{3} \quad \leq \quad |D_i| \quad < \frac{2n}{3}$$

because the difference between consecutive sizes of D_i is at most $n/3$.

We now show the decomposition.

- X is a forest zone. We partition X into the forest zone D_i and the context zone $X - D_i$. Both parts have between a third and two thirds the nodes.
- X is a context zone, which means that X is the difference $Y - Z$ of two forest zones. There are two cases to consider.

– D_i is a context zone, which means that $Z \subseteq D_i$. In this case we decompose X into two context zones D_i and $X - D_i$.

– D_i is a forest zone. In this case, the set $X - D_i$, which itself is not a zone, can be partitioned into at most three zones, all of which have at most one third of the nodes.

□

COROLLARY 2. *Let t be a forest. There exists exists a tree s, where each node x is labelled by a (forest or context) zone in t such that*

- *The root of s is labelled by the set of all nodes in t.*
- *Leaves of s are labelled by singletons.*
- *Let x be a non-leaf node of s, and let $x_1, \ldots, x_k$ be its children. Then $k \leq 4$, and the zones in the labels of $x_1, \ldots, x_k$ form a partition of the zone in the label of x.*
- *The depth of s is logarithmic in the size of X.*

PROOF. Apply Lemma 3 to the zone containing all nodes, and then recursively apply the same lemma to the resulting zones. Each time, the size of the zone decreases to at most $2/3$, so the process creates a tree of depth at most $\log_{3/2} n$. □

7.2 Incremental updates

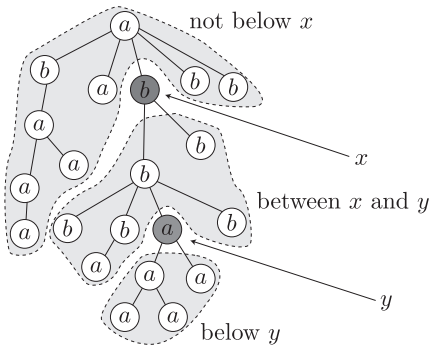
Using the hierarchical decomposition from Corollary 2, we can generalize to trees the algorithms from Section 3. The algorithm for incremental updates for tree languages is exactly the same as in the case of words. We just use the data structure given in Corollary 2.

7.3 Binary queries on forests

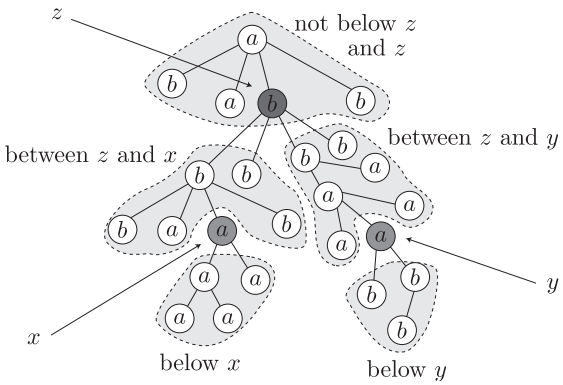
As in Section 3, we can use the hierarchical decomposition to evaluate binary queries. Most of the discussion in this section is devoted to defining regular binary queries for trees; once these are defined, a straightforward application of the hierarchical decomposition can be used to evaluate them.

In this section, we consider regular binary queries on trees (actually, on forests). Let x, y be two nodes in a forest t. There are four possible scenarios for their relative positions in the forest.

1. The first scenario is when x is an ancestor of y. In this case, the nodes of the forest can be partitioned into: x, y, two context zones and one forest zone, as in the picture below:



2. The second scenario is when x is a descendant of y. A decomposition similar to the one in the first scenario can be found.
3. The third scenario is when x and y are incomparable with respect to the descendant relation, but x comes first in document order. Let z be the closest common ancestor of x and y. In this case the complement of $\{x, y\}$ can be partitioned into three context zones and two forest zones, as in the following picture:



4. There is a degenerate variant of the third scenario, when x and y are descendants of different roots, and z does not exist. In this degenerate case when z does not exist, the uppermost context zone is empty.
5. The fifth scenario is when x and y are incomparable with respect to the descendant relation, but y comes first in document order. A decomposition similar to the one in the third scenario can be found.
6. The sixth scenario is the degenerate version of the fifth.

Suppose that α is a forest algebra morphism. Then type α-type of a pair of nodes (x, y) in a forest t consists of the following information:

- the labels of x and y;
- which scenario holds;
- for the appropriate scenario, the values assigned by α to the zones

A query is said to be recognized by α if there is a set F of α-types such that for every forest and pair of nodes (x, y), the pair (x, y) is selected by the query if and only if its α-type belongs to F. A query is called *regular* if it is recognized by some α. A similar definition makes sense queries of higher arities, such as three or four, but the number of scenarios grows with the arity of the query.

The above definition of regular queries coincides with more standard definitions of regular binary queries for trees or forests, such as queries defined by a formula of monadic second-order logic with two free variables.

THEOREM 7. *Let φ be a regular binary query on forests. There is an algorithm which:*

1. *on input forest t, builds a data structure in linear time;*
2. *using the data structure, answers in logarithmic time questions of the form: is a pair of nodes (x, y) selected by φ?*

PROOF. The same proof as for Theorem 3, except using the decomposition from Corollary 2. □

7.4 References

The first algorithm for incremental updates of regular tree languages is from [1]. The algorithm here is a slight improvement over [1], because for a tree with n nodes it runs in time $O(\log n)$ and not $O(\log^2 n)$.

8. REFERENCES

[1] Andrey Balmin, Yannis Papakonstantinou, and Victor Vianu. Incremental validation of xml documents. *ACM Trans. Database Syst.*, 29(4):710–751, 2004.

[2] Mikołaj Bojańczyk. Algebra for trees. In *Handbook of Automata Theory*. European Mathematical Society Publishing House. To appear.

[3] Mikołaj Bojańczyk. Factorization forests. In *Developments in Language Theory*, pages 1–17, 2009.

[4] Mikołaj Bojańczyk. and Paweł Parys. Efficient evaluation of nondeterministic automata using factorization forests. In *ICALP (1)*, pages 515–526, 2010.

[5] Thomas Colcombet. Factorization forests for infinite words and applications to countable scattered linear orderings. *Theor. Comput. Sci.*, 411(4-5):751–764, 2010.

[6] Gudmund Skovbjerg Frandsen, Johan P. Hansen, and Peter Bro Miltersen. Lower bounds for dynamic algebraic problems. *Inf. Comput.*, 171(2):333–349, 2001.

[7] Manfred Kufleitner. The height of factorization forests. In *MFCS*, pages 443–454, 2008.

[8] Frank Neven. Automata theory for xml researchers. *SIGMOD Record*, 31(3):39–46, 2002.

[9] Imre Simon. Factorization forests of finite height. *Theor. Comput. Sci.*, 72(1):65–94, 1990.

A Brief Survey of Automatic Methods for Author Name Disambiguation

Anderson A. Ferreira[1,2] Marcos André Gonçalves[2] Alberto H. F. Laender[2]

[1]Departamento de Computação
Universidade Federal de Ouro Preto
35400-000 Ouro Preto, Brazil

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
31270-901 Belo Horizonte, Brazil

{ferreira, mgoncalv, laender}@dcc.ufmg.br

ABSTRACT

Name ambiguity in the context of bibliographic citation records is a hard problem that affects the quality of services and content in digital libraries and similar systems. The challenges of dealing with author name ambiguity have led to a myriad of disambiguation methods. Generally speaking, the proposed methods usually attempt to group citation records of a same author by finding some similarity among them or try to directly assign them to their respective authors. Both approaches may either exploit supervised or unsupervised techniques. In this article, we propose a taxonomy for characterizing the current author name disambiguation methods described in the literature, present a brief survey of the most representative ones and discuss several open challenges.

1. INTRODUCTION

Several scholarly digital libraries (DLs), such as DBLP[1], CiteSeer[2], PubMed[3] and BDBComp[4], provide features and services that facilitate literature research and discovery as well as other types of functionality. Such systems may list millions of bibliographic citation records (here understood as a set of bibliographic attributes such as author and coauthor names, work and publication venue titles of a particular publication[5]) and have become an important source of information for academic communities since they allow the search and discovery of relevant publications in a centralized manner. Studies based on the DL content can also lead to interesting results such as topic coverage, research tendencies, quality and impact of publications of a specific subcommunity or individuals, collaboration patterns in social networks, etc. These types of analysis and information, which are used, for instance, by funding agencies on decisions for grants and for individual's promotions, presuppose *high quality content* [29, 31].

According to Lee et al. [31], the challenges to have high quality content comes from data-entry errors, disparate citation formats, lack of (enforcement of) standards, imperfect citation-gathering software, ambiguous author names, and abbreviations of publication venue titles. Among these challenges, *author name ambiguity* has required a lot of attention from the DL research community due to its inherent difficulty. Specifically, name ambiguity is a problem that occurs when a set of citation records contains ambiguous author names, i.e., the same author may appear under distinct names (synonyms), or distinct authors may have similar names (polysems). This problem may be caused by a number of reasons, including the lack of standards and common practices, and the decentralized generation of content (i.e., by means of automatic harvesting [30]).

To illustrate the problem, Table 1 shows a set of three citations $\{c_1, c_2, c_3\}$ so that each citation has its author names identified by $r_j, 1 \leq j \leq 16$. For each citation c_i, each name r_j is a reference to an author and has a list of attributes associated with it, such as, coauthor names (i.e., the list of references to other authors of the same citation), work title, publication venue title, publication year and so on. Examining Table 1, we see examples of synonyms and polysems, which, as mentioned before, are subproblems of the name ambiguity problem. Author names r_3 and r_{15} are examples of polysems where r_3 refers to "Ajay Gupta" from IBM Research India and r_{15} refers to "Aarti Gupta" from NEC Laboratories America, USA. Author names r_3 and r_7 are examples of synonyms. Both refer to "Ajay Gupta" from IBM Research India.

More formally, the name disambiguation task may be formulated as follows: Let $C = \{c_1, c_2, ..., c_k\}$ be a set of citation records. Each citation record c_i has a list of attributes which includes at least author names, work title and publication venue title. With each attribute in a citation is associated a specific value, which may have

[1]http://dblp.uni-trier.de
[2]http://citeseer.ist.psu.edu
[3]www.ncbi.nlm.nih.gov/pubmed
[4]http://www.lbd.dcc.ufmg.br/bdbcomp
[5]We use the terms "citation" and "citation record" interchangeably.

Table 1: Illustrative Example (Ambiguous Group of A. Gupta)

Citation Id	Citation
c_1	(r_1) S. Godbole, (r_2) I. Bhattacharya, (r_3) A. Gupta, (r_4) A. Verma. Building re-usable dictionary repositories for real-world text mining. CIKM, 2010.
c_2	(r_5) Indrajit Bhattacharya, (r_6) Shantanu Godbole, (r_7) Ajay Gupta, (r_8) Ashish Verma, (r_9) Jeff Achtermann, (r_{10}) Kevin English. Enabling analysts in managed services for CRM analytics. KDD, 2009.
c_3	(r_{11}) T. Nghiem, (r_{12}) S. Sankaranarayanan, (r_{13}) G. E. Fainekos, (r_{14}) F. Ivancic, (r_{15}) A. Gupta, (r_{16}) G. J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. HSCC, 2010.

several components. In case of the *author names* attribute, a component corresponds to the name of a single unique author and is a reference r_j to a real author. In case of the other attributes, a component corresponds to a word/term. The objective of a disambiguation method is to produce a function that is used to partition the set of references to authors $\{r_1, \ldots, r_m\}$ into n sets $\{a_1, \ldots, a_n\}$, so that each partition a_i contains (all and ideally only all) the references to a same author.

To disambiguate the bibliographic citations of a DL, first we may split the set of references to authors into groups of references whose values of the author name attribute are ambiguous. These are called ambiguous groups (i.e., groups of references having the value of the author name attribute with similar names). The ambiguous groups may be obtained by using blocking methods [37] which address scalability issues avoiding the need for comparisons among all references.

The challenges of dealing with name ambiguity in citation records have led to a myriad of disambiguation methods [3, 4, 7, 9, 15, 16, 20, 21, 22, 24, 26, 27, 33, 35, 38, 40, 41, 42, 43, 44, 46, 49]. One such a challenge is that, usually, only a minimum set of attributes is available to work with (in most case only author names and publication and venue titles). In any case, existing disambiguation methods usually attempt to either group citation records of the same author using some type of similarity between them or try to directly assign the citation records to their respective authors.

An early survey with some preliminary disambiguation methods is found in [28]. In that work, Klass classifies the methods into supervised or unsupervised ones and describes some methods published until 2006. However, the area has been very prolific in the last years, with many methods recently proposed. In this article, we propose a new taxonomy for characterizing the current methods for disambiguating author names and present a brief survey of some of the most representative ones.

This article is organized as follows. Section 2 proposes our taxonomy for characterizing the author name disambiguation methods. Section 3 presents an overview of representative author name disambiguation methods. A summary of characteristics of the methods is presented in Section 4. Section 5 discusses some open challenges in the author name disambiguation task. Finally, Section 6 presents our conclusions.

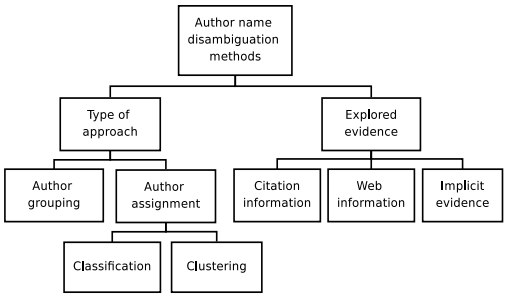


Figure 1: Proposed taxonomy

2. A TAXONOMY FOR AUTHOR NAME DISAMBIGUATION METHODS

This section presents a hierarchical taxonomy for grouping the most representative automatic author name disambiguation methods found in the literature. The proposed taxonomy is shown in Figure 2. The methods may be classified according to the main type of exploited approach: *author grouping* [4, 7, 9, 15, 16, 22, 24, 26, 27, 36, 38, 41, 42, 44, 45, 46, 35, 49], which tries to group the references to the same author using some type of similarity among reference attributes, or *author assignment* [3, 16, 20, 21, 43], which aims at directly assigning the references to their respective authors. Alternatively, the methods may be grouped according to the evidence explored in the disambiguation task: the citation attributes (only), Web information, or implicit data that can be extracted from the available information.

We should notice that in this survey we cover only automatic methods. Other types of method, such as manual assignment by librarians [39] or collaborative efforts[6], heavily rely on human efforts, which prevent them from being used in massive name disambiguation tasks. For this reason, they are not addressed in this article. There are also efforts to establish a unique identification to each author, such as the use of an Open Researcher Contributor Identification[7] (ORCID), but these are also not covered here.

Since the name disambiguation problem is not restricted to a single context, it is also worth noticing that several other name disambiguation methods, which exploit distinct pieces of evidence or are targeted at other applications (i.e., name disambiguation in Web search

[6]http://meta.wikimedia.org/wiki/WikiAuthors
[7]http://www.orcid.org

results), have been described in the literature [2, 12, 18, 48, 50]. However, a discussion of these methods is outside the scope of this article.

Finally, we should stress that the categories in our taxonomy are not completely disjoint. For instance, there are methods that use two or more types of evidence or mix approaches. In the next subsections, we detail our proposed taxonomy.

2.1 Type of Approach

As said before, one way to organize the several existing author name disambiguation methods is according to the type of approach they exploit. We elaborate this distinction further in the discussion below.

2.1.1 Author Grouping Methods

Author grouping methods apply a similarity function to the attributes of the references to authors (or group of references) to decide whether to group the corresponding references using a clustering technique. The similarity function may be predefined (based on existing ones and depending on the type of the attribute) [4, 7, 22, 36, 41], learned using a supervised machine learning technique [9, 24, 44, 45, 46], or extracted from the relationships among authors and coauthors, usually represented as a graph [15, 33, 35]. This similarity function is then used along with some clustering technique to group references of a same author, trying to maximize intra and minimize inter-cluster similarities, respectively.

Defining a Similarity Function

Here, a similarity function is responsible for determining how similar two references (or groups of references) to authors are. The goal is to obtain a function that returns high similarity values for references to the same author and returns low similarity values for references to different authors. Moreover, it is desirable that the similarity function be transitive. More specifically, let c_1, c_2 and c_3 be three citations, if c_1 and c_2 are very similar (according to the function) and c_2 and c_3 are also very similar, then c_1 and c_3 should have high similarity according to our function. Next, we discuss the ways to determine this similarity function.

Using Predefined Functions. This class of methods has a specific predefined similarity function $\mathcal{S}$ embedded in their algorithms to check whether two references or groups of references refer to the same author. Examples of such function $\mathcal{S}$ include [6]: the Levenshtein distance, Jaccard coefficient, cosine similarity, soft-TFIDF and others [6], applied to elements of the reference attributes. Ad-hoc combinations of such functions have also been used (e.g., in [4, 41])

These methods do not need any type of supervision in terms of training data but their similarity functions are usually tuned to disambiguate a specific collection of citation records. For different collections, a new tuning procedure may be required. Finally, not all the functions used in these methods are transitive by nature.

Learning a Similarity Function. Learning a specific similarity function usually produces better results, since these learned functions are directly optimized for the disambiguation problem at hand. To learn the similarity function, the disambiguation methods receive a set $\{s_{ij}\}$ of pairs of references (the training data) along a special variable that informs whether these two corresponding references refer to the same author. The pair of references, r_i and $r_j \in R$ (the set of references) are usually represented by a similarity vector $\vec{s_{ij}}$. Each similarity vector $\vec{s_{ij}}$ is composed of a set $\mathcal{F}$ of q features $\{f_1, f_2, \ldots, f_q\}$. Each feature f_p of these vectors represents a comparison between attributes $r_i.A_l$ and $r_j.A_l$ of two references, r_i and r_j.

The value of each feature is usually defined using other functions, such as Levenshtein distance, Jaccard coefficient, Jaro-Winkler, cosine similarity, soft-TFIDF, euclidean distance, etc., or some specific heuristic, such as the number of terms or coauthor names in common, or special values such as the initial of the first name along with the last names, etc.

The training data is then used to produce a similarity function $\mathcal{S}$ from R x R to $\{0, 1\}$, where 1 means that the two references do refer to the same author and 0 means that they do not. As mentioned before, methods relying in learning techniques to define the similarity function are quite effective in different collections of citations, but they usually need many examples and sufficient features to work well, which can be very costly to obtain.

Exploiting Graph-based Similarity Functions. The methods that exploit graph-based similarity functions for author name disambiguation usually create a coauthorship graph $G = (V, E)$ for each ambiguous group. Each element of the author name and coauthor name attributes is represented by a vertex v. The same coauthor names are usually represented by only a unique vertex. For each coauthorship (i.e., a pair of authors who publishes an article) an edge $\langle v_i, v_j \rangle$ is created. The weight of each edge $\langle v_i, v_j \rangle$ is related to the amount of articles coauthored by the corresponding author names represented by vertices v_i and v_j.

A graph-based metric (e.g., shortest path as in [33]) may be combined with other similarity functions on the attributes of the references to authors or used as a new feature in the similarity vectors.

Clustering Techniques

The author grouping methods usually exploit a clustering technique in their disambiguation task. The most used techniques include: 1) *partitioning* [23], which creates a pre-defined number k of partitions of the set of references to authors in an iterative process; 2) *hier-*

archical agglomerative clustering [23], which groups the references to authors in a hierarchical manner; 3) *density-based clustering* [23], in which a cluster corresponds to a dense region of references to authors surrounded by a region of low density (according to some density criteria) – references in regions with low density are considered as noise; and 4) *spectral clustering* [51], which corresponds to graph-based techniques that compute the eigenvalues and eigenvectors, the spectral information, of a Laplacian Matrix that, in the the author name disambiguation task, represents a similarity matrix of a weighted graph. In general, these clustering techniques rely on a "good similarity function" to group the references.

2.1.2 Author Assignment Methods

Author assignment methods directly assign each reference to a given author by constructing a model that represents the author (e.g., the probabilities of an author publishing an article with other (co-)authors, in a given venue and using a list of specific terms in the work title) using either a supervised classification technique [16, 20] or a model-based clustering technique [3, 21].

Classification. Methods in this class assign the references to their authors using a supervised machine learning technique. More specifically, they receive as input a set of references to authors with their attributes, the *training data* $\mathcal{D}$, consisting of examples or, more specifically, references for which the correct authorship is known. Each example is composed of a set $\mathcal{F}$ of m features $\{f_1, f_2, \ldots, f_m\}$ along with a special variable called the *author*. This *author* variable draws its value from a discrete set of labels $\{a_1, a_2, \ldots, a_n\}$, where each label uniquely identifies an author. The training examples are used to produce a disambiguation function (i.e., the disambiguator) that relates the features in the training examples to the correct author. The *test set* (denoted as $\mathcal{T}$) for the disambiguation task consists of a set of references for which the features are known while the correct author is unknown. The disambiguator, which is a function from $\{f_1, f_2, \ldots, f_m\}$ to $\{a_1, a_2, \ldots, a_n\}$, is used to predict the correct author for the references in the test set. In this context, the disambiguator essentially divides the records in $\mathcal{T}$ into n sets $\{a_1, a_2, \ldots, a_n\}$, where a_i contains (ideally all and only all) the references in which the ith author is included.

These methods are usually very effective when faced with a large number of examples of citations for each author. Another advantage is that, if the collection has been disambiguated (manually or automatically), the methods may be applied only to references of the new citations inserted into the collection by simply running the learned model on them. Although successful cases of the application of these methods have been reported, the acquisition of training examples usually requires skilled human annotators to manually label references. DLs are very dynamic systems, thus manual labeling of large volumes of examples is unfeasible. Further, the disambiguation task presents nuances that impose the need for methods with specific abilities. For instance, since it is not reasonable to assume that examples for all possible authors are included in the training data and the authors change their interesting area over time, new examples need be insert into training data continuously and the methods need to be retrained periodically in order to maintain their effectiveness.

Clustering. Clustering techniques that attempt to directly assign references to authors work by optimizing the fit between a set of references to an author and some mathematical model used to represent that author. They use probabilistic techniques to determine the author in a iterative way to fit the model (or estimate the parameters in probabilistic techniques) of the authors. For instance, in the first run of such a method each reference may be randomly distributed to an author a_i and a function, from a set of features $\{f_1, f_2, \ldots, f_m\}$ to $\{a_1, a_2, \ldots, a_n\}$, is derived using this distribution. In the second iteration, this function is used to predict the author of each reference and a new function is derived to be used in the next iteration. This process continues until a stop condition is reached, for instance, after a number of iterations. Two algorithms commonly used to fit the models in disambiguation tasks are Expectation-Maximization (EM) [11] and Gibbs Sampling [19].

These methods do not need training examples, but they usually require privileged information about the correct number of authors or the number of author groups (i.e., group of authors that publish together) and may take some time to estimate their parameters (e.g., due to the several iterations). Additionally, these methods may be able to directly assign authors to their references in a new citations using the final derived function.

2.2 Explored Evidence

In this section, we describe the kinds of evidence most commonly explored by the disambiguation methods.

Citation Information. Citation information are the attributes directly extracted from the citations, such as author/coauthor names, work title, publication venue title, year, and so on. These attributes are the ones commonly found in all citations, but usually are not sufficient to perfectly disambiguate all references to authors. Some methods also assume the availability of additional information such as emails, addresses, paper headers, which is not always available or easy to obtain, although if existent, they usually help the process (a lot!).

Web Information. Web information represents data retrieved from the Web that is used as additional information about an author publication profile. This informa-

tion is usually obtained by submitting queries to search engines, based on the values of citation attributes and the returned Web pages are used as new evidence (attributes) to calculate the similarity among references. The new evidence usually improves the disambiguation task. One problem is the additional cost of extracting all the needed information from the Web documents.

Implicit Evidence. Implicit evidence is inferred from visible elements of attributes. Several techniques have being implemented to find implicit evidence, such as the latent topics of a citation. One example is the Latent Direchlet Location (LDA) [5] that estimates the topic distribution of a citation (i.e, LDA estimates the probability of each topic given a citation). This estimated distribution is used as new evidence (attribute) to calculate the similarity among references to authors.

2.3 Evaluation Metrics

Although not part of our taxonomy, one important point to understand the discussion that follows is the evaluation metrics that are used by each proposed method in their experimental evaluations. The most used metrics are: *accuracy*, which is basically the proportion of correct results among all predictions; the traditional metrics of *precision*, *recall*, and *F1*, commonly used for information retrieval and classification problems[8]; *pairwise F1*, a variation of F1 that considers pairs of citations correctly assigned to the same author (or not); *Cluster F1*, that calculates precision and recall of the correct clusters (i.e., the clusters that contain all and only all the references to an author); the *K* metric [7], a combination of purity (a pure cluster contains citations of only one author) and fragmentation of clusters (fragmentation occurs when the production of one author is split into one or more clusters); B-Cubed [1], that calculates precision and recall for each reference to an author; and MUC [1]. In this last metric, recall is calculated by summing up the number of elements in the ground-truth clusters minus the number of empirical clusters (obtained with the method) that contain these elements and dividing this by the total of elements minus the number of theoretical clusters. Precision is calculated similarly.

3. OVERVIEW OF REPRESENTATIVE METHODS

In this section, we present a brief overview of representative author name disambiguation methods which fall under one or more of the categories of the proposed taxonomy. Our main focus here is on those methods that have been specifically designed to address the name ambiguity problem in the context of bibliographic citations, since they are more related to the scope of this work. In the next subsections, we describe each method under the category we consider that best fits it. We notice that most of the described methods explore citation information in the disambiguation task. Thus, we leave to Subsection 3.3 the discussion of those methods that use additional evidence.

[8]In this last case, the authors are considered as classes and the correct assignments need to be known a priori.

3.1 Author Grouping Methods

Using Predefined Functions. Han et al. [22] represent each reference as a feature vector where each feature corresponds to an element of a given instance of one of its attributes. The authors consider two options for defining the feature weights: TFIDF and NTF (Normalized Term Frequency), being NTF given by $ntf(i,d) = freq(i,d)/maxfreq(i,d)$ where $freq(i,d)$ refers to the feature frequency i within the record d, and $maxfreq(i, d)$ refers to the maximal term frequency of feature i in the record d. The authors propose the use of K-way spectral clustering with QR decomposition [51] to construct clusters of references to the same author. To use this clustering technique, the correct number of clusters to be generated needs to be informed. The K-way spectral clustering method represents each reference as a vertex of an undirected graph and the weight of an edge represents the similarity between the attributes associated with the connected references. K-way spectral clustering splits the graph so that records that are more similar to each other will belong to the same cluster. This method was evaluated using data obtained from the Web and DBLP. Experimental results achieved 63% of accuracy in DBLP and up to 84.3% in the Web collection.

An algorithm for collective entity resolution (i.e., an algorithm that uses only disambiguated coauthor names when disambiguating an author name of a citation) that exploits attribute elements (i.e., attribute values present in the citation records) and relational information (i.e., authorship information between entities referred in the citations records) is proposed in [4] by Bhattacharya and Getoor. The authors propose a combined similarity function defined on attributes and relational information. As the initial step, they create clusters of disambiguated references verifying if two references have at least k coauthor names in common (they used only the author names in their experiments but they mention that other attributes may be used). The experiments were performed using soft-TFIDF, Jaro-Winkler, Jaro and Scaled Levenshtein metrics for name attributes, and Common Neighbours, Jaccard coefficient, Adamic-Adar similarity and Higher-order neighbourhoods metrics for relational attributes. The authors exploit a greedy agglomerative strategy that merges the most similar clusters in each step. The collections used in the experiments were: a subset of CiteSeer containing machine learning documents, a collection of high energy physics

publications from arXiv[9] and BioBase[10] that contains biological publications from Elsevier. The method obtained around 0.99 of F1 in the CiteSeer and arXiv collections and around 0.81 in the BioBase collection.

In [41], Soler proposes a new distance metric between two citations, c_i and c_j, (or clusters of citations) based on the probability of these publications having terms and author names in common. The proposed algorithm creates clusters of articles using the proposed metric and summarizes the clusters by means of a representative citation that includes the distance from it to the others. It groups the citations whose distances among them is minimum using as evidence author names, email, address, title, keywords, research field, journal and publication year. The final decision whether two candidate clusters belong to the same author or not is given by a specialist. Soler presents some illustrative cases of clusters obtained by using the proposed metric with records extracted from ISI-Thomson Web of Science database[11].

In [7], Cota et al. propose a heuristic-based hierarchical clustering method for author name disambiguation that involves two steps. In the first step, the method creates clusters of references with similar author names that share at least a similar coauthor name. Author name similarity is given by a specialized name comparison function called *Fragments*. This step produces very pure but fragmented clusters. Then, in the second step, the method successively fuses clusters of references with similar author names according to the similarity between the citation attributes (i..e., work title and publication venue) calculated using the cosine measure. In each round of fusion, the information of fused clusters is aggregated (i.e., all words in the titles are grouped together) providing more information for the next round. This process is successively repeated until no more fusions are possible according to a similarity threshold. The authors used pairwise F1 and K metrics on collections extracted from DBLP and BDBComp to evaluate the method and obtained around 0.77 and 0.93 for K in DBLP and BDBComp, respectively. An extension of this method that allows the name disambiguation task to be incrementally performed is presented in [10].

Learning a Similarity Function. In [44], Torvik et al. propose to learn a probabilistic metric for determining the similarity among MEDLINE records. The learning model is created using similarity vectors between two references containing features resulting of the comparison between the common citation attributes along with medical subject headings, language, and affiliation of two references. They also propose heuristics for generating training sets (positive and negative) automatically. In a subsequent work [45], Torvik and Smalheiser extend the method by including additional features, new ways of automatically generating training sets, an improved algorithm for dealing with the transitivity problem and a new agglomerative clustering algorithm for grouping records. They estimate recall around 98.8%, that only 0.5% of the clusters have mixed references of different authors (purity), and that only in 2% of the cases the references of a same authors are split into two or more clusters (fragmentation).

In [24], Huang et al. present a framework in which a blocking method is first applied to create blocks of references to authors with similar names. Next DBSCAN, a density-based clustering method [14], is used for clustering references by author. For each block, the distance metric between pairs of citations used by DBSCAN is calculated by a trained online active support vector machine algorithm (LASVM), which yields, according to the authors, a simpler and faster model than the standard support vector machines. The authors use different functions for each different attribute, such as the edit distance for emails and URLs, Jaccard similarity for addresses and affiliations and soft-TFIDF for names. The authors have applied their framework to a manually annotated dataset with 3,335 citation records and 490 distinct authors. Experiments were performed with pairs of references in which the disambiguator informs whether two references are of the same author or not. They obtained 90.6% in terms of pairwise F1. It should be noticed that these results were obtained by exploiting additional sources of evidence such as the headers of papers obtained from CiteSeer.

In [9], Culotta et al. aim to learn a score function to be applied to the disambiguation result, such that higher scores correspond to the more correct disambiguations. Instead of calculating the score using pairs of references, the authors propose a score function that considers all references in a cluster together, with the goal of maximizing the result of the score function in the resulting disambiguation. To learn this function, they propose a training algorithm that is error-driven, i.e., training examples are generated from incorrect predictions in the training data and ranked, i.e., the classifier uses a ranking of candidate predictions to tune its parameters. The authors evaluated two loss functions to tune the parameters, Ranking Perceptron [17] and Ranking MIRA [8]. The experimental evaluation used two collections extracted from DBLP (one which is called Penn, because disambiguation was performed manually by students from Penn State University) and other from the Rexa[12] Digital Library. As evaluation metrics, they used pairwise F1, MUC and B-Cubed [1]. As evidence, they ex-

[9] http://arxiv.org

[10] http://www.elsevier.com/wps/find/bibliographicdatabasedescription.cws_home/600715/description#description

[11] http://isiknowledge.com

[12] http://rexa.info

ploited features such as first and middle names of the authors, number of coauthors in common, rarity of the last name, similarity between work titles, e-mails, affiliations and publication venue titles as well as the minimum, maximum and average values for real-valued features, among several others. They also used a greedy agglomerative clustering technique to group the references. Ranking Perceptron generated the best results in DBLP and Penn, with 0.52 and 0.86 of pairwise F1, respectively. Ranking MIRA generates the best result on the other DBLP collection with 0.931 of pairwise F1.

Treeratpituk and Giles [46] propose a learned similarity function for author name disambiguation in the MEDLINE digital library. The authors exploit a large feature set obtained from MEDLINE metadata, similar to that proposed in [44]. The authors also use similarity vectors to learn the similarity function using a Random Forest classifier. They compare the use of Random Forests with decision trees, support vector machines, naïve Bayes and logistic regression to learn the function to be used along with some clustering technique (left unspecified). They also investigate the performance of subsets of the features capable of reaching good effectiveness. The authors obtain almost 96% of accuracy in their experiments by exploiting this large set of features.

Exploiting Graph-based Similarity Functions. In [35], On et al. address synonym in the group entity resolution problem (i.e., a reference to a person associated with a group of items, e.g., an author with a list of publications) by proposing an approach that uses the quasi-clique graph-mining technique for exploiting, besides simple textual similarities, "contextual information" extracted from the group items' attributes (e.g., the citation attributes) as additional evidence. This contextual information is obtained constructing a graph for each group to represent relationships between the author names (i.e., references) and the attribute values (e.g., co-authors). This graph is then superimposed on the pre-built graph constructed using the entire set of author names. Using this contextual information, the authors also propose a graph-based distance function based on common quasi-clique between the graphs of two entities (i.e., references). They compared their graph-based function (distQC) with Jaccard, TF-IDF and IntelliClean functions [32] by measuring the precision and recall at the top k most similar references using three collections extracted from ACM[13], BioMed (a dataset of medical publications) and IMDb. On average, the experiments show an improvement of 63%, 83% and 46% over Jaccard, TFIDF and IntelliClean functions in terms of precision at top-k records returned by their algorithm in ACM. Similar results were obtained for the other collections.

In [33], Levin and Heuser propose social network metrics that, along with string metrics, generate match functions to verify whether two references represent the same author. These functions were used in (very small) collections extracted from Cora[14], BDBComp and DBLP. The authors construct a graph with two kinds of vertices: one represents a reference to an author occurring in a citation and the other represents the citation itself; and two kinds of edges: one links the reference to the citation and the other links the vertices that share the same author name value. The authors obtained in their experiments around 95%, 82% and 95% of F1 in versions of Cora, BDBComp and DBLP, respectively.

In [15], Fan et al. propose the GHOST (GrapHical framewOrk for name diSambiguaTion) framework. GHOST solves the polysem problem using only the coauthor name attribute in five steps. In the first one, GHOST represents a collection as a graph G=(V, E), where each vertex $v \in V$ represents a reference to be disambiguated and each undirected edge (v_i, v_j) represents a coauthorship whose label S_{ij} is a set of citations coauthored by v_i and v_j. In the second step, GHOST identifies the valid paths eliminating the invalid ones between two nodes, i.e., a path that contains a subpath $v_i S_{ik} v_k S_{kj} v_j$ where S_{ik} is equal to S_{kj} and both have only one citation. In the third step, GHOST creates a matrix representing similarities between the vertices. For this, the authors propose a new similarity function based on the formula that calculates the resistance of a parallel circuit. In the fourth step, the Affinity Propagation clustering algorithm [13] is used to group the references to the same author. Finally, in the last step, GHOST makes use of user feedback to improve the results. Experimental evaluation was performed in collections extracted from DBLP and MEDLINE. GHOST obtained on average 0.86 and 0.98 of pairwise F1 in DBLP and MEDLINE, respectively.

3.2 Author Assignment Methods

Classification. In [20], Han et al. propose two methods based on supervised learning techniques that use coauthor names, work titles and publication venues as evidence for assigning a reference to its author. The first method uses naïve Bayes (NB), a generative statistical model frequently used in word sense disambiguation, to capture all writing patterns in the authors' citations. The second method is based on Support Vector Machines (SVMs), which are discriminative models basically used as a classifier [34]. An important difference between the two techniques is that a NB model requires only positive examples to learn about the writing patterns whereas SVMs require both positive and negative examples to learn how to identify the author. Both methods have been evaluated with data taken from the Web and DBLP. Experimental results show that, on average,

[13] http://portal.acm.org

[14] http://www.cs.umass.edu/ mccallum/code-data.html

using all attributes, the SVM-based method was more accurate (accuracy=95.6%) than the NB method (accuracy=91.3%) for the Web collected dataset while for the DBLP dataset the NB method performed better (SVM accuracy was 65.4% while NB's was 69.1%).

In [47], Veloso et al. propose SLAND, a disambiguation method that infers the author of a reference by using a supervised rule-based associative classifier. The method uses author names, work title and publication venue title attributes as features and infers the most probable author of a given reference r_i using the confidence of the association rules $\mathcal{X} \rightarrow a_i$ where $\mathcal{X}$ only contains features of r_i. The method also works on demand, i.e., association rules to infer the correct author of a reference are generated in the moment of disambiguation. The method is also capable of inserting new examples into the training data during the disambiguation process, using reliable predictions, and detecting authors not present in the training data. Experiments were conducted in two collections extracted from DBLP and BDBComp and the proposed method outperformed representative supervised (SVM and NB) considering the Micro and Macro F1 metrics. In the DBLP and BDBComp collections, the (Micro) F1 values were 0.911 and 0.457, respectively. To reduce the cost of obtaining training data, this method was extended [16] to become self-trained, i.e., it is now capable of producing its own training examples using (test) references to be disambiguated. Initially, the method extracts pure clusters of references by exploiting highly discriminative features, such as co-author names. The most dissimilar clusters are then selected to represent training examples for their authors. Next, the references in the rest of clusters are classified according to these training examples. In the experiments with the same collections, the self-trained method outperformed by far KWAY and SVM-DBSCAN and the associative method was the best choice for classifying the remaining test references not incorporated into the training data when compared to SVM and NB.

Clustering. In [21], Han et al. present an unsupervised hierarchical version of the naïve Bayes-based method for modeling each author. The authors assume that each citation is generated by a mixture of K authors. They then calculate the probability of a citation record c_m given an author a_i P($c_m|a_i$) using the probability of each attribute of this record given such author, in a hierarchical way. To estimate the parameters, the authors use the Expectation Maximization algorithm [11] aiming to maximize the likelihood of the citation records. The method obtained on average 54% and 58% of accuracy on data extracted from DBLP and the Web, respectively.

In [3], Battacharya and Getoor extend the generative model Latent Dirichlet Allocation (LDA) and propose a probabilistic model for collective entity resolution that uses the cooccurence of the references to authors in each work to determine the entities jointly, i.e., they use the disambiguated references to disambiguate other references in the same citation. In their model, the authors associate an attribute v_a, that contains the author name in the citation, with each author a. They assume that each citation is constructed choosing their authors from an author group (i.e., a group of authors that publish some article together) distribution. That is, initially a distribution that determines the probability of each author group having a specific author chosen to write the article is selected. Next using this distribution, the authors and a variation of their names are chosen for this citation. The proposed method receives as input only an approximation of the number of author groups in the collection. Experiments were performed using citations extracted from CiteSeer and arXiv reaching up to 0.99 and 0.98 respectively of pairwise F1.

In [43], Tang et al. propose a probabilistic framework based on Hidden Markov Random Models (HMRF) for the polysem subproblem. In this work, the authors use author names, work title, publication venue title, publication year, abstract and bibliographic references as content-based evidence and relationships between citations as structure-based evidence for disambiguating author names. Each relationship represents the fact that two citations were published in the same publication venue, have a coauthor name in common, cite each other, have distinct coauthor names that were coauthors in another citation, or have some specific user-provided constraint in common. Content and structure-based evidence are modeled as feature functions (used to represent the similarity between two citations by their content or relationships) which are then incorporated into a HMRF used to estimate the weights of the feature functions and to assign the citations to their authors. The authors also use Bayesian Information Criterion to estimate the number of authors of the collection. Experimental evaluation was performed on citations extracted from ArnetMiner[15]. Pairwise F1 values were 0.888 and 0.805 when the method uses the correct number of authors and when it estimates this number, respectively.

3.3 Using Additional Evidence

Web Information. In [26], Kanani et al. present two approaches for author name disambiguation that gather additional evidence from the Web. They construct a graph in which each vertex corresponds to a reference to an author and the edges are weighted with values that represent the probability of the two vertices (i.e., references) being the same author. This weight is initially calculated using the citation attributes. In the first approach, they use the result of searches submitted to a

[15] http://arnetminer.org

Web search engine for the work titles of citation records of the corresponding references to authors to change the weight of the edge between two references. In the second one, they use one of the returned pages of the search as a new type of vertex in the graph (web vertex), adding new edges from this new vertex to each previously existing reference vertex, indicating the probability of the reference and the web page belonging to the same author. In both cases, a maximum entropy or logistic regression model is learned for a pair of references a_i and a_j and the weight of the edge $\langle a_i,a_j \rangle$ is given by the probability of the corresponding references refer to the same author minus the probability of these references refer to the different authors. DBLP, Penn and the Rexa collections were used in their experiments. Using the results of searches to Google to change the weight of the edges their method obtain around 0.905, 0.877 and 0.918 of accuracy and around 0.886, 0.814 and 0.747 of pairwise F1 in the DBLP, Rexa and Penn collections, respectively. The method that uses the returned Web pages as vertices in the graph was run only with DBLP, producing 0.882 of accuracy and 0.903 of pairwise F1.

In [49], Yang et al. address the author name ambiguity problem using topics and correlations found on the Web. They determine the topics of the citation from venue information using an extraction algorithm based on association rules in order to create a topic association network. They also use the Web for retrieving publication pages of authors or coauthors to be disambiguated. Then, they create a similarity function making use of an SVM classifier on the top of all these features. The authors represent the references to authors as vertices in a graph and the similarity function is used to create the edges between vertices. Their clustering technique removes a bridge edge when each resulting connected component has at least a given number of vertices. They tested their approach on the collection constructed by Han et al. [20] and obtained an increase of accuracy around 66% (0.75 of accuracy) when compared to the use of citations without topics and Web correlations.

In [27], Kang et al. exploit coauthorship information using a Web-based technique that obtains other (implicit) coauthors of the reference to be disambiguated. They submit a pair of author names of a same citation as a query to Web search engines to retrieve documents containing both author names and then extract new names found in these documents as new implicit coauthors of this pair. The authors measure the similarity between two references by counting the number of coauthors in common and use the single-link agglomerative clustering technique [25] to group the references to the same author. They used a collection of citations published in Korean during 1999-2006 that has only the polysem problem obtaining around 0.85 of pairwise F1.

In [38], Pereira et al. also exploit Web information to disambiguate author names. Their method attempts to find Web documents corresponding to curricula vitae or Web pages containing publications of a single author. It works in three steps. The first step receives a list of citations whose references must be disambiguated and, for each citation, submits a query containing data from its attributes to a Web search engine. It then inserts the top-m documents in the answer set into a set $\mathcal{D}$ of documents. The second step selects the documents in $\mathcal{D}$ that contain publications from a given author. The third step groups the reference to authors whose citations occur in a same document in a hierarchical manner, i.e., if citations of two ambiguous references occur in the same Web document, these citations are considered as belonging to the same author and are fused in a same cluster. The experimental evaluation, performed using DBLP data, obtained on average 0.80, 0.76 and 0.14 of K, pairwise F1 and cluster F1 metrics, respectively.

Implicit Evidence. In [42], Song et al. propose a two-step unsupervised method. The first step uses Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA) to assign a vector of probabilities of topics to each citation. The PLSA and LDA proposed by them introduce a variable for persons (authors) in the generative model, that does not exist in general generative models. The second step considers the distributions of the probability of topics with respect to citations as a new attribute for name disambiguation. The authors use the Levenshtein distance to measure the similarity between two names. When two names are considered similar, they use the probability vectors of two corresponding citations and the Euclidean distance to merge the citations of the same authors. The authors compared their method with a greedy agglomerative clustering, K-way spectral clustering and LASVM + DBSCAN on citations extracted from CiteSeer and personal names on the Web. Their experiments demonstrate that their method, when faced with a lot of citation information is more effective than the baselines obtaining on average around 0.911 and 0.936 of pairwise F1 on the Web and CiteSeer collections, respectively.

In [40], Shu, Long and Meng extend the Latent Dirichlet Allocation model (LDA) for obtaining the topic distribution of each citation by adding the assumption that every topic is a Dirichlet distribution over all author names, that each document is a mixture of topics, and that each topic is a Dirichlet distribution over all the words. They train a classifier (C4.5 and SVMs) based on the similarity on topics, coauthor names, title and venue, as well as on the minimum distance between coauthor names, to predict whether two references are of the same author or not. The authors attempt to solve the problem of name ambiguity by trying to solve first

the polysem problem and then the synonymy. They use K-way spectral clustering to split the references into k sets, one for each author, in order to deal with the polysem problem. Next, they compare two sets of references of authors whose names have a distance below a given threshold and count the number of citations from these two sets which are assigned to the same author by the classifier. This value is divided by the total number of pairs of those two sets and if the result is greater than a given threshold they are merged. The authors show the effectiveness of their method by applying it to data extracted from DBLP. For the polysem problem the precision an recall were over 0.9 for the most ambiguous groups while for the synonym problem the precision was around 0.99 and recall equals to 0.917.

4. SUMMARY OF CHARACTERISTICS

In this section, we present an overview of the characteristics found in the described author name disambiguation methods, summarized in Tables 2 and 3.

Among the collections used to evaluate the methods we have: (1) versions of CiteSeer, DBLP, BDBComp, ArnetMiner, and Rexa containing computer science publications; (2) arXiv that contains citations from high energy physics publications; (3) BioBase, containing citations from biological publications; (4) MEDLINE and BioMed with data from biomedical publications; (5) ISI-Thomson with publications from several knowledge areas; (6) Cora, which is constituted of duplicated citations in Computer Science and person names extracted from the Web; and (7) IMDb with data about actors.

The majority of the described methods [4, 7, 9, 15, 22, 24, 26, 27, 33, 35, 38, 40, 41, 42, 45, 46, 49] try to disambiguate references to authors by using a similarity function to indicate whether two references refer to the same author instead of directly assigning the corresponding author to each reference, as in [3, 16, 20, 21, 43, 47].

Some of these methods receive the correct number of authors in the collection as input ([15, 21, 22]) ors this number corresponds to the number of authors in the training data [20]. Other methods, such as those proposed in [3], [43] and [16], try to estimate this number.

Almost half of the methods [3, 4, 7, 15, 16, 20, 21, 22, 33, 35, 40] uses at most the three main citation attributes: author names, work title and publication venue title, as evidence. These attributes are the most commonly found in citation records, constituting in most cases the hardest situation for disambiguation. Few methods [26, 27, 38, 49] exploit additional evidence such as emails, addresses, paper headers etc., which are not always available or easy to obtain.

Tables 2 and 3 also summarize the evaluation metrics used by each method as well as the type of subproblem (i.e., synonym, polysem, or both) addressed.

5. OPEN CHALLENGES

There are several open challenges that need to be addressed in order to produce more reliable solutions that can be employed in a production mode in real digital libraries. Below we discuss some of them.

Very Little Data in the Citations. In most cases we have only the basic information about the citations available: author (coauthor) names, work and publication venue titles, and publication year. Furthermore, in some cases author names contain only the initial and the last surname and the publication venue title is abbreviated. New strategies that try to derive implicit information (e.g., topics) or gather additional information from the Web are promising in this scenario.

Very Ambiguous Cases. Several methods exploit coauthor-based heuristics, by explicitly assuming the hypotheses that: (i) very rarely ambiguous references will have coauthors in common who have also ambiguous names; or (ii) it is rare that two authors with very similar names work in the same research area. These hypotheses work in most cases, but when they fail, the errors they generate are very hard to fix. For example, in the case of authors with Asian names, the first hypothesis fails more frequently than for authors with English or Latin names.

Citations with Errors. Errors occur in citation data which are sometimes impossible to detect. The methods need to be tolerant to such errors.

Efficiency. With the high amount of articles being published nowadays in the different knowledge areas, the solutions need to deal with the problem efficiently. Few proposed methods have this explicit concern.

Different Knowledge Areas. As we have seen, most of the collections used to evaluate the methods are related to Computer Science. However, other knowledge areas (e.g., Humanities, Medicine) may have different publication patterns (e.g., publications with a sole author or with a lot of coauthors) causing additional difficulties for the current generation of methods.

Incremental Disambiguation. Ideally disambiguation should be performed incrementally as new citations are incorporated into the DL, since it is not reasonable to assume that the whole DL should be disambiguated at each new load. However, most, if not all, methods ignore this fact. A promising solution is presented in [10].

Author Profile Changes. It is common that the research interests of an author change over time. This can happen due to new collaborations, change in research group or institution, natural evolution of a research field, etc. These changes cause modifications in the model representing the author profile causing difficulties for the methods. A possible solution may involve retraining, but determining when to retrain is a challenge. However, this issue has been largely ignored by all methods.

New Authors. The methods should be capable of iden-

Table 2: Summary of characteristics - Author grouping methods

Method	Similarity function	Clustering technique	Evidence	Collections	Evaluation metric	Subproblem	# of authors
Bhattacharya and Getoor [4]	Common neighbours, Jaccard, Adamic/Adar and Higher-order neighbourhoods	Agglomerative	Author name	CiteSeer, arXiv and BioBase	F1	Both	Unknown
Cota et al. [7]	Fragment comparison and cosine	Agglomerative	Citation attributes	DBLP and BDBComp	Pairwise F1 F1 and K	Both	Unknown
Culotta et al. [9]	Error-drive and hank-based learning	Agglomerative	All of each collection	DBDL and Rexa	Pairwise F1, MUC and B-Cubed	Both	Unknown
Fan et al. [15]	graph-based	Affinity Propagation	Author names	DBLP and MEDLINE	Pairwise F1	Polysem	Unknown
Han et al. [22]	Cosine	Spectral clustering	Citation attributes	DBLP and Web	Accuracy	Both	Known
Huang et al. [24]	Learned using LASVM	DBScan	First page of the articles	CiteSeer	Pairwise F1	Both	Unknown
Kanani, McCallum and Pal [26]	Learned using maximum entropy or logistic regression	Partitioning	Citation attributes and Web pages	DBLP, Penn and Rexa	Accuracy and pairwise F1	Both	Unknown
Kang et al. [27]	Heuristic	Agglomerative	Author names and Web pages	Korean citations	F1 and under/ over-clustering error	Polysem	Unknown
Levin and Heuser [33]	Social network metrics	-	Citation attributes	DBLP, Cora and BDBComp	F1	Both	Unknown
On et al. [35]	Quasi-clique	-	Citation/Movie attributes	ACM, BioMed and IMDb	Ranked recall and precision	Synonym	Unknown
Pereira et al. [38]	Heuristic	Agglomerative	Citation attributes	DBLP	Pairwise and cluster F1 and K	Both	Unknown
Shu, Long and Meng [40]	Learned using C4.5/SVMs and edit distance	Spectral and agglomerative clustering	Citation attributes	DBLP	Pairwise F1	Both	Known
Soler [41]	Probabilistic metric	Agglomerative	Citation attributes, email, address, keywords and research field	ISI-Thomson	-	Both	Unknown
Song et al. [42]	Levenshtein and Euclidean distance	Agglomerative	Citation attributes and latent topics (LDA/PLSA)	CiteSeer and Web	Pairwise and cluster F1	Both	Unknown
Torvik and Smalheiser [45]	Learn a probabilist metric	Agglomerative	MEDLINE metadata	MEDLINE	Recall	Both	Unknown
Treeratpituk and Giles [46]	Learned using random forest classifier	-	MEDLINE metadata	MEDLINE	Accuracy	Both	Unknown
Yang et al. [49]	Learned using SVM	Partitioning	Citation attributes, topics and Web pages	DBLP	Accuracy, precision and recall	Both	Unknown

Table 3: Summary of characteristics - Author assignment methods

	Method	Technique	Attributes	Collections	Evaluation metric	Subproblem	# of authors
Classification	Ferreira et al. [16]	Associative classifier	Citation attributes	DBLP and BDBComp	Pairwise F1 and K	Both	Estimated
	Han et al. [20]	SVM and naïve Bayes classifiers	Citation attributes	DBLP and Web	Accuracy	Both	Known
	Veloso et al. [47]	Associative classifier	Citation attributes	DBLP and BDBComp	F1	Both	Estimated
Clustering	Battacharya and Getoor [3]	LDA with Gibbs sampling	Author names	CiteSeer and arXiv	F1	Both	Estimated
	Han et al. [21]	Hierarchical naïve Bayes with EM	Citation attributes	DBLP and Web	Accuracy	Both	Known
	Tang et al. [43]	Hidden Markov Random Fields	Citation attributes	ArnetMiner	Pairwise F1	Polysem	Estimated

tifying references to new ambiguous authors who do not have citations in the DL yet. Only one of the reported methods [47] has explicitly addressed this issue.

6. CONCLUSIONS

This article presented a brief survey on author name disambiguation methods. We proposed a taxonomy to classify the methods and provided an overview of the most representative ones. Some patterns became clear. The majority of the surveyed methods perform disambiguation by comparing citation records using some type of similarity function. This function, which can be predefined or learned specifically for the disambiguation task, is directly applied to the citation attributes, which can be enhanced with additional information retrieved from the Web or inferred from the own citation attributes (e.g., topics). A few other methods disambiguate by directly assigning the citation records to their authors using supervised and unsupervised machine learning techniques. Some open problems were also discussed.

One major gap in the field is the lack of direct comparisons among the methods under the same circumstances: e.g., same collections (e.g., many methods used different versions of collections such as DBLP), same computational environment, same experimental design. This is probably due to the lack of standard collections like those provided by the TREC competitions. Moreover, the few comparisons that exist involve at most three or four methods and were performed in static scenarios. In fact, there is no study about how these methods would perform in a real-word scenario of a dynamic and living digital library. These issues along with the open problems previously discussed are in our opinion what should guide the research efforts for developing new author name disambiguation methods in the near future.

Acknowledgments

This research is partially funded by InWeb (MCT/CNPq/FAPEMIG grant 573871/2008-6), CAPES, CNPq, and FAPEMIG.

7. REFERENCES

[1] A. Bagga and B. Baldwin. Algorithms for scoring coreference chains. In *LREC*, pages 563–566, 1998.
[2] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *WWW*, pages 463–470, 2005.
[3] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *SDM*, 2006.
[4] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM TKDD*, 1(1), 2007.
[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.
[6] W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, pages 73–78, 2003.
[7] R. G. Cota, A. A. Ferreira, M. A. Gonçalves, A. H. F. Laender, and C. Nascimento. An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations. *JASIST*, 61(9):1853–1870, 2010.
[8] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *JMLR*, 3:951–991, 2003.
[9] A. Culotta, P. Kanani, R. Hall, M. Wick, and A. McCallum. Author disambiguation using error-driven machine learning with a ranking loss function. In *IIWeb*, 2007.
[10] A. P. de Carvalho, A. A. Ferreira, A. H. F. Laender, and M. A. Gonçalves. Incremental unsupervised name disambiguation in cleaned digital libraries. *JIDM*, 2(3):289–304, 2011.
[11] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38, 1977.
[12] C. P. Diehl, L. Getoor, and G. Namata. Name reference resolution in organizational email archives. In *SDM*, pages 70–91, 2006.
[13] D. Dueck and B. J. Frey. Non-metric affinity propagation for unsupervised image categorization. In *ICCV*, 2007.
[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.
[15] X. Fan, J. Wang, X. Pu, L. Zhou, and B. Lv. On graph-based name disambiguation. *JDIQ*, 2:10:1–10:23, 2011.
[16] A. A. Ferreira, A. Veloso, M. A. Gonçalves, and A. H. F. Laender. Effective self-training author name disambiguation in scholarly digital libraries. In *JCDL*, pages 39–48, 2010.
[17] Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.
[18] C. Galvez and F. de Moya Anegón. Approximate personal name-matching through finite-state graphs. *JASIST*, 58(13):1960–1976, 2007.
[19] T. Griffiths and M. Steyvers. Finding scientific topics. *The National Academy of Sciences*, 101(1):5228–5235, 2004.
[20] H. Han, C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *JCDL*, pages 296–305, 2004.
[21] H. Han, W. Xu, H. Zha, and C. L. Giles. A hierarchical naive Bayes mixture model for name disambiguation in author citations. In *SAC*, pages 1065–1069, 2005.
[22] H. Han, H. Zha, and C. L. Giles. Name disambiguation in author citations using a k-way spectral clustering method. In *JCDL*, pages 334–343, 2005.
[23] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2005.
[24] J. Huang, S. Ertekin, and C. L. Giles. Efficient name disambiguation for large-scale databases. In *ECML-PKDD*, pages 536–544, 2006.
[25] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.
[26] P. Kanani, A. McCallum, and C. Pal. Improving author coreference by resource-bounded information gathering from the web. In *IJCAI*, pages 429–434, 2007.
[27] I.-S. Kang, S.-H. Na, S. Lee, H. Jung, P. Kim, W.-K. Sung, and J.-H. Lee. On co-authorship for author disambiguation. *Inf. Process. Manage.*, 45(1):84–97, 2009.
[28] V. C. Klaas. Who's who in the world wide web: Approaches to name disambiguation. Master's thesis, Diplomarbeit, LMU München, Informatik, 2007.
[29] A. H. F. Laender, M. A. Gonçalves, R. G. Cota, A. A. Ferreira, R. L. T. Santos, and A. J. C. Silva. Keeping a digital library clean: new solutions to old problems. In *DocEng*, pages 257–262, 2008.
[30] C. Lagoze and H. V. de Sompel. The open archives initiative: building a low-barrier interoperability framework. In *JCDL*, pages 54–62, 2001.
[31] D. Lee, J. Kang, P. Mitra, C. L. Giles, and B.-W. On. Are your citations clean? *Comm. ACM*, 50(12):33–38, 2007.
[32] M.-L. Lee, T. W. Ling, and W. L. Low. IntelliClean: a knowledge-based intelligent data cleaner. In *KDD*, pages 290–294, 2000.
[33] F. H. Levin and C. A. Heuser. Evaluating the use of social networks in author name disambiguation in digital libraries. *JIDM*, 1(2):183–197, 2010.
[34] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
[35] B.-W. On, E. Elmacioglu, D. Lee, J. Kang, and J. Pei. Improving grouped-entity resolution using quasi-cliques. In *ICDM*, pages 1008–015, 2006.
[36] B.-W. On and D. Lee. Scalable name disambiguation using multi-level graph partition. In *SDM*, pages 575–580, 2007.
[37] B.-W. On, D. Lee, J. Kang, and P. Mitra. Comparative study of name disambiguation problem using a scalable blocking-based framework. In *JCDL*, pages 344–353, 2005.
[38] D. A. Pereira, B. A. Ribeiro-Neto, N. Ziviani, A. H. F. Laender, M. A. Gonçalves, and A. A. Ferreira. Using web information for author name disambiguation. In *JCDL*, pages 49–58, 2009.
[39] C. L. Scoville, E. D. Johnson, and A. L. McConnell. When A. Rose is not A. Rose: the vagaries of author searching. *Medical Reference Services Quaterly*, 22(4):1–11, 2003.
[40] L. Shu, B. Long, and W. Meng. A latent topic model for complete entity resolution. In *ICDE*, pages 880–891, 2009.
[41] J. M. Soler. Separating the articles of authors with the same name. *Scientometrics*, 72(2):281–290, 2007.
[42] Y. Song, J. Huang, I. G. Councill, J. Li, and C. L. Giles. Efficient topic-based unsupervised name disambiguation. In *JCDL*, pages 342–351, 2007.
[43] J. Tang and *et al*. A unified probabilistic framework for name disambiguation in digital library. *TKDE*, 24(6):975–987, 2012.
[44] V. I. Torvik , M. Weeber, D. R. Swanson, and N. R. Smalheiser. A probabilistic similarity metric for Medline records: A model for author name disambiguation. *JASIST*, 56(2):140–158, 2005.
[45] V. I. Torvik and N. R. Smalheiser. Author name disambiguation in MEDLINE. *ACM TKDD*, 3(3):1–29, 2009.
[46] P. Treeratpituk and C. L. Giles. Disambiguating authors in academic publications using random forests. In *JCDL*, pages 39–48, 2009.
[47] A. Veloso, A. A. Ferreira, M. A. Gonçalves, A. H. F. Laender, and W. Meira Jr. Cost-effective on-demand associative author name disambiguation. *Inf. Process. Manage.*, 48(4):680– 697, 2012.
[48] Q. M. Vu, T. Masada, A. Takasu, and J. Adachi. Using a knowledge base to disambiguate personal name in web search results. In *SAC*, pages 839–843, 2007.
[49] K.-H. Yang, H.-T. Peng, J.-Y. Jiang, H.-M. Lee, and J.-M. Ho. Author name disambiguation for citations using topic and web correlation. In *ECDL*, pages 185–196, 2008.
[50] M. Yoshida and *et al*. Person name disambiguation by bootstrapping. In *SIGIR*, pages 10–17, 2010.
[51] H. Zha and *et al*. Spectral relaxation for K-means clustering. In *NIPS*, pages 1057–1064, 2001.

Tamer Özsu Speaks Out

On journals, conferences, encyclopedias and technology

by Marianne Winslett and Vanessa Braganholo

Tamer Özsu
http://cs.uwaterloo.ca/~tozsu/

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 Providence, site of the 2009 SIGMOD and PODS conference. I have here with me Tamer Özsu, who is a Professor of Computer Science and a University Research Chair at the University of Waterloo in Canada. He is also the Director of the David Cheriton School of Computer Science there[1]*. Before joining the University of Waterloo, Tamer was at the University of Alberta, also in Canada. Tamer's research interests lie in multimedia databases, distributed data management and XML. Tamer received the SIGMOD Contributions Award in 2006, and he is an ACM Fellow and a former member of the VLDB Endowment board, as well as a former chair of ACM SIGMOD. Tamer's PhD is from The Ohio State University.*

[1] His term ended in July 2010.

So Tamer, welcome!

Thank you.

Tamer, you worked on object-oriented databases for several years and they never really "caught on". Do you think there will be a resurgence of interest in object-oriented databases?

Well, object-oriented databases never caught on as a replacement for relational systems. But a lot of the technology that was developed within the context of object-oriented database research made its way into object-relational systems. So, it is all there, but in a different context. The pure object systems do not exist. But object-orientation has actually made a comeback on the modeling tools, so there is a lot enterprise modeling tools, etc. that are object-oriented. I have recently listened to a talk by industry folks where the description was on the tools that people are now developing to take object-oriented models that have been developed and map them into relational systems. These tools are exceptionally complicated, and the hoops that you have to jump through are incredible, so if we had actually done object-orientation properly, then we probably wouldn't have had to jump through these. So it has never died, they exist in various forms, and in the modeling tools, they seem to be very popular and used today.

What do you think about this conference versus journal debate?

Hot topic. I am on the record as actually having stated that we have abused the conference system tremendously. I suspect the number of people who have read it are very small, but one of my SIGMOD chair notes in SIGMOD Record (I think it was in 2003) was on journals versus conferences. Perhaps the most relevant phrase I used there was that "we have done a wonderful job in convincing tenure and promotion committees of the value of conferences, but now we have to convince ourselves of the value of journals". So I think all of our attempts to "fix" the conferences, increasing paper submissions, avoiding bad reviews, unreliable reviews, variability in the reviews, etc. are really the outcome of pushing the system far beyond what they were designed to do. We have ignored the journals. That is actually our fault. We should have never ignored the journals to the extent that we have. A lot of the arguments that are made against journals no longer hold: long review cycles, etc. The top two journals in data management, which are VLDB Journal and TODS, have first round review cycles that are now shorter than conference review times, from submission to appearance. Reviews are probably more solid; they are more detailed, there is a cycle where you can respond to reviews and so on. I think we should revisit the value of journals.

So suppose we take our journals very seriously, what will our conferences look like then?

Well, I think one of the things is that conferences were supposed to be places for early dissemination of very new ideas. Right now there are far from that. A large chunk of the papers are basically incremental work that are very valuable, but they could just as well go into archival journals. The conferences should have their paper length reduced to about 8 pages, and they should actually focus on new ideas. The conference papers don't have to have all the i's dotted and t's crossed. They don't all have to have experiments. We should actually turn the conference to what they were supposed to be.

There is no reason that we need to have this many conferences either. We could actually have a few ones where the major ideas are presented, and the deeper technical content of the development of those ideas could lead to journal work.

But then, assuming that there is a certain amount of work trying to get out and be seen, how would the journals cope with the enormous increase in submissions?

Well, I think if you really compare it, there are a number of things going on there. If you compare the number of conference and journal submissions in computer science to some more established disciplines, there is no question that our numbers are far below what, for example, the chemists or the physicists or even some of the engineering disciplines produce. So, they manage to process it all within the context of a journal culture. The typical review cycles that are given in some of these journals in other sciences is about two to three weeks. You need to actually do the review and return it, now granted that the papers are shorter. But if we really change the culture, then our journal papers do not have to have 30% more content over already long conference papers to get to a size where it is really arduous to do the review. We can actually review them much faster. Now a lot of the journals are online, you can actually change it from print version to an on-demand print, so the page limit and the cost associated with page limits are no longer an issue. So there are ways of addressing this, ways that other disciplines have actually figured out, so that we can actually figure it out. We don't have to be identical to the other disciplines, we need to find our own culture, but clearly over-relying on conferences as the major outlet and the final paper publication outlet has stressed our system tremendously.

If we had actually done object-orientation properly, then we probably would not have had to jump through these [object-relational mappings].

So what is PVLDB, and how does it fit into this discussion?

PVLDB, to the extent that I understand it, it is an initiative of the VLDB Endowment Board led by Jag (H. V. Jagadish). Since I am no longer on VLDB Endowment Board, I don't know all the internal details, but to the extent that I know, at the steady state, the model that they wish to have is that paper submissions will be done to PVLDB. It is the Proceedings of the VLDB, which is an online journal, and is also included in ACM digital library. At the steady state, what will happen as I understand it, is that you will actually submit papers to PVLDB. And they guarantee a first round turnaround time of 1 month, so it is going to be very quick. But you will have the typical journal review process, which is you get the review, if they ask for revisions, you can do the revisions, respond to reviewers, comments, argue with the reviewers if you wish, and then you go through that more elaborate and probably better system of peer review. And then the papers will be accepted, they will appear as journal publications, in PVLDB, and the VLDB conference

program committee will select some of these papers for presentation. So if you have a paper published in PVLDB, you will get a journal publication, plus you make a presentation at the VLDB conference. That is the steady state that they are moving towards. Going from here to there is going to be tricky.

Right now[2] what is actually happening is that you submit papers the PVLDB, or you submit papers to the VLDB conference normally. There are no conference proceedings, the conference proceedings appear as PVLDB issues. And a certain number of papers that appear in PVLDB that have not been submitted to the conference directly, are selected by the conference program committee for presentation as well. But this is a transition as I understand, talking to Jag. The steady state will be that the journal will be the base, and the conference will be basically presentation overlay on top of that base.

So how does PVLDB relate to the VLDB Journal?

They are separate. I think, as I understand it, the VLDB Endowment is treating the VLDB Journal as a regular, classical journal. So you could have published papers at conferences before, you can increase [their content], the sizes are longer in the VLDB Journal than in PVLDB, and it has a regular [paper submission] track. What the relationship is going to be over the long run between PVLDB and VLDB Journal I suppose remains to be worked out. I don't exactly know what the VLDB Endowment Board discussions were on that one.

We have done a wonderful job in convincing tenure and promotion committees of the value of conferences, but now we have to convince ourselves of the value of journals.

So I see a lot of encyclopedias springing up, and I see that you've written chapters for some of them, I've written chapters for some of them… who is going to be reading all of these encyclopedias?

Well, I have not only written chapters for some of them, I am actually with Ling Liu, editing a major Encyclopedia in database systems[3]. But other than the one that we're actually working on, all of the others are really handbooks. So you have chapters devoted to them. Ours is somewhat different in that it is really a reference encyclopedia. There are regular entries, which are limited in the number of words, and we have definitional entries which are even shorter, some on the order of a page or so. So we really wanted to get a reference encyclopedia, more along the lines of Encyclopedia

[2] This refers to the state as of June 2009.
[3] The encyclopedia was published in 2009: Encyclopedia of Database Systems, Springer, 2009, ISBN 978-0-387-35544-3.

Britannica and so on, which are not really lengthy chapters, but shorter encyclopedia entries, with cross references to other entries. And the idea is that you never actually read an encyclopedia from cover to cover. There are some of us who actually like reading encyclopedias, but generally, you don't, you refer to them. So they are reference works. That is the context in which we operated in this forthcoming encyclopedia of database systems. It has about 1300 entries, will be about 5,000 pages published in 5 volumes. But, more importantly, it will be published online, so you can access it online and it will also be indexed in Michael Ley's DBLP, so you will be able to quickly get to the articles. So, it is a reference work.

So, who is publishing it?

It is Springer. Springer is publishing it.

So I'll pay to read it.

Well, if your university is a subscriber to the Springer link, and almost all North American Universities that I know of are subscribers, you get it free. All the authors who contributed entries, and there are about 880 them, get free online access.

So how long is the article on say, query optimization, you say they are length limited?

They are all length limited. I think the regular articles are 3,000 to 4,000 words, and the definitional entries are about 1000 words, so they are really relatively short.

Can you talk about the tradeoffs between stuffing XML data into a relational DBMS versus building a native XML DBMS?

Well in many of these, there are good arguments to make it either way. I mean, we have invested about 30 years of research into relational systems, they are very mature. You can cover a lot of distance by using the relational engines to support these complex data types, and so on. So there are arguments, but I've always (this goes back to my object oriented work as well) preferred to work in the pure object and the pure XML mode, just because whatever you do, some of those techniques will find their way into the other one as well. And I don't necessarily think that we need to actually have a one pony game, where we tie everything into the relational engine, and you do everything. It almost reminds me of a Turing machine. We know that anything we can compute, we can compute with a Turing machine, but none of us are actually talking about programming Turing machines.

Well, maybe we are, because of virtualization!

Well, I mean, computer science is all about abstractions, building abstractions, and virtualization is an abstraction that hides certain things that you do underneath. But the issue is that there are parts of the relational engine, in the relational technology, that are definitely relevant in the XML world, or any other world. But that doesn't necessarily mean that you basically tie everything to the relational [technology] and figure out how to map a complex data type to a flat tabular structure and do the processing there. I think we need to separate the technology that we

developed for relational systems, from the relational systems themselves. There is room for the technology obviously to play a role in declarative querying, optimization techniques, etc, and even some of the optimization techniques that we use can certainly carry over to how we support these more complex data types. But we don't need to actually force everything into the relational engine. I think that probably is not the right thing to do. Even the most recent discussion of column store versus row store, is an indication that not all applications need to be forced into a single architecture for data management. I think that we need to be able to break out and figure out what are the really critical essentials that we build into very [small], perhaps micro-kernel type engines, and what needs to be left out for customization, for different types of applications, and different types of data types.

Interesting. So have you built a system like that?

Well, we started building one in the object world, and we actually went quite a bit, and then we got distracted with other research interests. We started in the XML world as well, but by the time we were doing the XML world, I really did not have the energy to build up a big implementation group anymore. I did that in the '90's and it was really a lot of fun, but I didn't want to repeat that one more time. So we really never pushed it to a reasonable prototype on the object side. We did have internal prototypes that we actually fooled around with for research purposes. But we did a little bit of it.

So benchmarks: what do we need for XML?

Well I think there are benchmarks that have actually been developed. We did one benchmark for XML called XBench, that started in collaboration with IBM Toronto Labs, and then we kind of took it on our own and went further. The fundamental point of our benchmark was that we did not actually want to just say, well, "what are some interesting queries, and what are some interesting XML structures that we should test these systems on", but we tried to go out and actually find actual customer data, and IBM helped quite a bit. They looked at their customer data, we didn't see the data, but at least the characterization. "What is going on in terms of the types of XML data that people develop, and what type of applications are we seeing"? So we came up with a taxonomy of the types of XML data that we were seeing, and then we developed a family of benchmarks that really had their roots on the statistical characteristics of the data that we were seeing in real life, for the most part. There were parts of the taxonomy for which we could not find the data, for which we went and looked at XML use cases, and other things. I think the important thing in the benchmarks is basically being able to defend the choices in the benchmark that you are making.

… not all applications need to be forced into a single architecture for data management.

So did you get a query workload too?

Yes, but our query workload at the time when we did this, which was very early 2000's, there weren't actually that many applications being built, so the query workloads are really a distillation of, a careful analysis of XML use cases that were reported as part of the XML standardization efforts. We looked at it and said "okay, what classes of workloads are these use cases representing", and work from that.

One of your colleagues have asked me to ask you when the 3rd edition of your textbook Principles of Distributed Database Systems will be out, and why is it taking so long.

A fair question and a touchy question! It is going to be out soon, how soon, I don't know[4]. The third edition is a major rewrite, and includes substantial new material. Every chapter has been reworked, but also there is a lot of new material that we didn't fit into previous chapters: replication, peer to peer systems, work data management, data integration, in a much fuller sense than we had before, etc. We have one chapter to write and two chapters to revise and then we'll be done. So, soon! The reason it took this long is because I took over the directorship of the school, and it is just time.

I understand that you used to be active in politics.

Yes, I was, when I was at university, and shortly after that.

And where were you?

That was in Turkey. I was a fairly left wing radical, more interested in that than in school things.

So what made you become more interested in CS?

Well, I think I was always interested in the CS part. During my undergrad degree, and actually my first master's in industrial engineering, I was always interested in CS. I kept taking courses in CS. Even when my degree advisor told me I could not take more CS courses, I kept taking them, so I graduated with extra [credits]. I was the type of student who did very well on the topics that I liked, and just barely survived on the topics that I didn't care about. So I was always interested in CS, but the specific interest in databases goes back to about 1976, when I was doing a part time masters, and working at the Turkish postal administration on a problem which basically can be the directory problem, the 411 system. You call up and you actually ask for a number, and we were supposed to be designing a system, but the system needed to be able to be queried using different keys, and we were struggling on how, what data structures to use, and how to lay it out so you could actually query it multiple ways. I was part time taking a graduate course, and part of the course was databases. The first book, Date's first edition, had just come out. We had a visiting professor from US who was giving a course, and light bulbs when on. I said "that is the solution to the problem that we were doing", and I got hooked up on databases. Back in '76.

So is that the approach you actually used your directory problem?

[4] The book came out on March, 2011.

Well, we didn't because there were no [database] systems then. I mean we used a general approach, but we couldn't use the system. You still had the index sequential stuff, so you had to build multiple indexes over the data, etc. But the general idea of how to lay it out so you could actually do it was it. Shortly thereafter, I started working for a United Nations project in Turkey, and I remember going to my first conference. It was the second VLDB conference in 1976 in Brussels… That's how long ago I got involved in databases.

How many students of yours have picked up your hobby of collecting pens?

I'm sorry to say, none. Although, we gave a fountain pen as a gift to one of my former students recently, so we are working on them, but so far, as far as I know, none of them have picked it up.

Well then, if they don't like it, why do you like it, why do you collect pens?

Well, I actually don't know why I like it. I think there is tremendous esthetics in fountain pens. They are very simple devices on the face of it. But they are really very sophisticated. Somebody actually wrote a whole monograph on the physics of fountain pens. The capillary action, why, what happens, how the air goes in and the ink replaces air, etc. You could get hooked on them for scientific reasons. Mine wasn't for scientific reasons, I just loved the esthetics of it, and I loved the feel of it when I write. I don't always use a computer to take notes. I use the old pen and paper approach. It has grown from an interest to a sickness.

You are going to be in this business for 30 plus years, so if you don't enjoy it, don't do it.

Oh, a sickness? How many do you have?

I think right now I have about 600.

That's a lot of pens. Do you insure your collection?

No, I don't. And I have some of them on my webpage. And my wife keeps asking me when we should expect somebody to see the webpage and then storm our house and steel them.

Actually, I didn't see them on your webpage. On your homepage I saw photos of your wife, son, dog, and your motorcycle, no pens.

Actually, if you go one below that, there is a link to my pens, so you missed it. It is there.

So why does that motorcycle rate above the pens?

This actually came from a talk I was preparing, and they said "we want you to talk about what you do outside of work". So I said, well, I spend time with my family, I walk my dog, these are things I enjoy, I collect pens, I had the link, and I love to ride my motorcycle, so I actually added them there so they appear. So the motorcycle is another passion.

Do you have an interesting story about one of your pens?

Well, every pen has an interesting story…

Oh, just one! Just one!

I'm going to have to think about it. Either the pen is interesting, or the purchase is interesting. One of the interesting stories is we were in a taxicab in Taipei in 1995 Data Engineering with my student, coming back from a museum. We were on a bridge, and I saw the glimpse of a pen shop, a hundred meters down at the other end of it, and we stopped the taxicab and went down and I bought the pen. My student could not believe that I could actually spot the pen shop that far away, about 50 meters below the bridge, a hundred meters on the other side. But us pen collectors have an eye for these things.

I guess so! Do you have any words of advice for fledgling or midcareer database researchers?

The word of advice is good luck. It is really far more stressful than when we started. Our expectations for hiring and tenure have grown to be what one might actually legitimately call ridiculous stage right now. If you take the acceptance rates and you look at what we expect, a PhD needs to start publishing a paper, or at least submitting papers, in their second term, which really makes you question whether that is possible. But you know, the pendulum swings in this business, and it will. I think a fundamental issue is, do what really attracts you, I think that is probably very cliché, but you are going to be in this business for 30 plus years, and if you don't enjoy it, don't do it.

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

Well, that is probably tough to say, it is almost like choosing which of your children are your favorites, so it is probably quite difficult. But I think the work that we did, even though we talked about it not making it in the main stream, the work that we did on object orientation was the most enjoyable because it really spanned fairly theoretical to system type of work to fairly pragmatic work, where we built query optimizers and tested them, etc. That and the multimedia image database work that I did in the second half of the 90's were end-to-end projects that we actually started from the architecture and the models all the way to languages and implementation. So those were probably quite interesting projects.

If you magically had enough extra time to do one additional thing at work that you are not doing now, what would it be?

Magically finding time is probably the right word. I'd love to do a lot more reading, which I'm not able to do right now.

I think you're supposed to say that you will finish your 3rd edition!

I'm trying not to think of that one! But that will get done in the next 6 months.

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

One thing that I wish I was better at was on more formal aspects. I can actually do certain things, but I'm not that good in the theoretical side of databases. And there are lots of interesting problems that I can skirt, but I cannot dig in there. So I wish I were actually a better theoretician to be able to tackle those.

Thank you very much for talking with me today.

Thank you for the opportunity.

Erich Neuhold Speaks Out

on industry research versus academic research, funding projects, and more

by Marianne Winslett and Vanessa Braganholo

Erich Neuhold
http://cs.univie.ac.at/mis-team/infpers/Erich_Neuhold/

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are in Indianapolis, site of the 2010 SIGMOD and PODS conferences. I have here with me Erich Neuhold, who is a professor at the University of Vienna. Until 2005 he was the director of the Fraunhofer Institute for Integrated Publication and Information Systems in Darmstadt, and a professor at the Technical University of Darmstadt. Erich has also worked for IBM and HP, both in Europe and in the US. Erich is an IEEE Fellow and a Fellow of the Gesellschaft fuer Informatik. Erich's PhD is from the Technical University of Vienna. So, Erich, welcome!

Thank you very much for having me!

What does it mean to have an integrated publication and information system?

I think in order to explain, I will add a little background. When I took over the directorship of the Fraunhofer Institute (formerly a GMD Institute), I felt that, coming from the database area, we

needed to add the human to this whole approach. In this situation, the human is a person that uses information, and then maybe creates new information. This is a cycle, and that's what integration here means: it is the cycle of locating information, processing it, finding the human user, creating new information, depositing it again. That is what we call the integration of information in this publication and information cycle. So that gave rise to the name, and we got stuck with it in reality, and couldn't change it because at some point publication was not anymore the right word, and the Web came along, and you couldn't call publishing what you now do with the information. But we got stuck with it, we had a brand name "IPSI" and we didn't want to change that, so we stayed with it, never finding a new paradigm for it.

Well, you had all these different groups. You were mentioning this really big Institute with 120 people total and a $10M/year budget. So what was it like to manage something that big?

Before I started with this institute, I was at HP as the director of one of their research labs. I had also about 60 people working for me. So anyway, that provided experience. I mean, if you step from, let's say, a small research group with 3 or 4 PhD students, to manage 120 people, some of them are older than you, and some of them are the same age, and some are really young, then this needs adjusting your behavior. But I felt it as a challenge, and I took up to that challenge when I took the job, and I think it went quite successful.

I'm actually a new director of an institute myself, so maybe I should ask you: do you have words of advice for me, and for the other people out there who are taking on these new roles?

"We distributed all our business cards. Everybody was interested". And my question was, "how many business cards did you collect"?

I think one has to look for contact with the people. I mean, in some ways, it is a little different from industry. When I was a director in industry, for example, it was not a good style to go down and work with researchers, because of my management team. I had two lines of management between me and them. They would come to me and say "don't you trust that we can do it?", whereas in a research community, this is different because as a professor, I was essentially also the PhD advisor of those younger people. And of course as a PhD advisor, you have to look for personal contact, and not delegate it to your management, because that would not really work. So this is a very different behavior I found between management in industry research and management in academic research.

What do you think of the way all this XML stuff has turned out?

I think when semi-structured data came along, let's say documents were put into databases (let's call it that way), I found (talking with publishers and other people) that actually storing the structure of these documents in the sense of XML or SGML at that time, was important. Then, of course, HTML came along with the Web, and proved that this concept was important, and became successful, I believe, because it was simpler than SGML. XML has a lot of SGML and HTML, but it is complicated. It is simpler than the first one, but more complicated than HTML. For that reason, I believe that it was a steeper step for people to go in this direction, and that is why XML wasn't as successful as I believed originally. I thought it would essentially take over the Web.

Well, I guess a long time ago, people thought object oriented databases would play this role, and then the relational vendors just added object oriented features. So is it the same way with XML, or did something different happen there?

I belong to the people that started out with relational databases, and then got very enthusiastic with object oriented databases, because in order to talk about entities, you shouldn't flatten them out into a table. You want to combine the attributes of different properties together, interconnect entities and also add manipulation functions. So I thought that was the way to go… And it was at a similar time when C++ came along in the programming world, and developed in about the same speed as it did in the database world. But then, object orientation in databases wasn't so successful. First of all, there were these strong vendors of relational database systems already, so there was steep competition. But I believe also that, in my opinion, the first commercial object oriented database systems never really worked well. They didn't really have the performance, the scaling, and things like that. And so people got disappointed with the vendors behind it and the established RDBS vendors pushing pressure on them said "there is never going to be any better [product than relational databases]". A self-fulfilling prophecy...

Do you believe in the Semantic Web?

No. I mean, this has to be qualified. The Semantic Web, in some way, is a vision. A vision in which many of us didn't believe in from the very beginning, because we felt it being too ambitious. You will not be able to describe all the semantic information that is hidden in documents, in their structures, and in any context information, in the way it was assumed by the semantic Web people, in order to be able to make automatic deductions and all kinds of other things. For a long time I have worked in data integration and as a curator of databases, and terminology discrepancies and all kinds of other problems have not been solved. So it was quite clear they are not going to solve them in a Semantic Web, and they didn't. Despite that some of their algorithms made sense or made contributions… For me, it is like in artificial intelligence. Artificial intelligence always had such high goals. And then it didn't succeed, and then funding stopped, and everybody kind of said they were not successful. But if you look a little more carefully, a little deeper, they actually made many contributions. There are quite a number of fields where their algorithms, like machine learning algorithms, have moved into systems. Supervised learning and unsupervised learning have played a role in many other applications. But of course, we are not recreating human intelligence with those machines.

Have the Web standards bodies played an important role in the creation of the Web as we see it today? The W3C…

I would say definitely not at the beginning. Tim Berners-Lee himself, he was working at CERN at the time when essentially the Web was created. He and his colleague Robert Cailliau actually wanted to build an SGML engine, but their manager – who told me the story – advised them "you need two years and three people to build that, but I'll give you half a year, and you two do it". So they had to cut down on SGML, simplified it and developed HTML as a hypertext system. There was no standard for doing it. It was just a simple code that allowed exchanging documents in the high energy physics domain, but they were very successful. And of course interfaces and tools like Mosaic and other things, made it useful for everybody. Again, Mosaic was not built on any standards or anything. Later on, when the field became a little more mature, and many other people joined in, I think standards started making more sense. If you don't have standards, for example, if you look at an HTML document, how to visualize it? You have many choices. So in that way, Mosaic was an ad-hoc standard. People just keep following whoever has the first successful approach. But then later you have to have some control of that. I think, then some of the standard things are very valuable, but you shouldn't go too early, because otherwise you will restrict the development.

How is it different to do research in the US versus in Europe?

This is a difficult question, but I will try to answer it. I think for example, research in Europe has similarities now within the universities at both places. When I started out as a professor, a long time ago, a professor in Germany had a number of research assistants, a number of teaching assistants, provided by the universities. So essentially, you had a budget and you didn't need to go out for project money or industry things. That changed. Of course, even then ambitious professors would always go for additional money. Starting out I had 3 researchers and 2 technicians from the University in Stuttgart, but in the end I had a team of 15 or 20 people. So I was able to have large projects. But the pressure was different; you didn't have to do it in order to be successful. So, that gave you an advantage, because you had a larger number of people, but you were free to do with them whatever you liked. There was no funding available in distributed databases when I started working on it. I started working on it because somebody in the institute bought about 10 PDP 11 computers and then we didn't know what to do with them, because the guy who was supposed to do something with them left. So they were sitting around with 64K addressing space, and as I was in the database field, I said "wouldn't it be good to distribute a database over these small machines, and utilize these machines in a distributed system"? There was nobody who would say that was a great idea, at least not at that time.

It is amazing to think of doing something significant with 64K of memory.

Oh yeah, but that was the time. I mean, I was talking to Nicholas Wirth at some point and he built the personal computer LILITH. We discussed, and he said "oh, 64K of memory is plenty for writing programs! Nobody writes larger programs". He was at Zurich, at ETH at that time. And I told him "this is not enough, I work in databases, and where do you store the data"? And he said "oh, you bring them in as you need them". And then a year later he came back to me, and said, "You were right, and I need a larger memory". But it was not because of the data. It was

because of the visualization he wanted to do on his screen. The machines were slow, so you didn't do graphics on the fly, you prepared the screen in the memory, and he needed as much memory as he had (64k) just for the higher resolution screen. So this was a very interesting observation.

Coming back; on the other side, I feel that having had NSF projects in the United States, and EU projects in Europe, the overhead in Europe is much higher and slows down research behavior.

When you say overhead, do you mean like management overhead?

Yeah, yeah, administrative overhead.

Is that because of the meetings?

This might be because of the multi-national teams, and of course, the idea behind it is that cooperation over Europe should be encouraged. On the other side, you travel a lot, the behavior, knowledge and culture of the different nationalities are different, and you spend a lot of time in meetings arguing about things which you would much better solve in your own home place. My institute had many researchers and in some of the areas we were involved in, we could have done the whole project much more efficiently ourselves, but we wouldn't have gotten any money for it without international partners, whereas in the United States, as an institute, I probably would have gotten the money.

The first commercial object oriented database system never really worked…

You've moved back and forth between academic and industrial research. What leads you to make those changes?

Opportunities. I mean, I was in industry, I worked for IBM at the beginning, and then in IBM, I moved into the field of databases. Then when I moved back to Europe, after being in IBM (I was in New York at that time), computer science in Germany was building up. Some friend of mine, whom I knew from earlier days, told me "Oh, I became a professor. Don't you want to become a professor too"? So I just applied, and I got 2 offers, one in Darmstadt and one in Stuttgart. In the end I took the Stuttgart position. Amazing enough, because you mentioned my Darmstadt association, I went to Darmstadt 20 years later. So it was Stuttgart, and I left IBM, and I built up a team there, and then I got contacted by Joel Birnbaum, I think he was vice-president of HP, whom I knew when he was the head of computer research of IBM in New York. I had spent some sabbatical leave there, so he knew me, and he sent me e-mail where he asked "do you know somebody who could lead a research group as research director in my team at HP in the database field"? I thought he meant me! And I applied. And it turns out that he didn't mean me because when he knew me in IBM, I was actually working on compilers. I even wrote a book on

compilers, so he thought I was a language person, and was very surprised, but he hired me anyway. This was kind of the reason I switched the second time into industry. And then I stayed in Palo Alto, and I enjoyed it very much working for HP. But I had a slight problem with my family, because they didn't want to come. I could have tried to force them but without ever applying, I got offers from Europe, as a professor again. So essentially, as a test I asked all that I wanted from the universities. If they gave it to me, I would take the job. If they didn't do it, I'd stay with HP. They gave me everything I asked for and I went back to Europe.

So there is one more switch to Fraunhofer, right?

Yeah, yeah, right. Actually, I went to Vienna as a professor from the United States. The Austrian president came to California and told me "we need people like you in Austria". So I got tricked into that, since when I came to Austria others there told me "we don't really need you here", but more polite, of course. As a result, I was not so happy, and I started looking around again. Soon, some other colleagues knew that I was looking around. I looked into the United States, looked at HP actually, as I thought to go back and even got an offer. But I also looked at academia at that time, and GMD came to me with an institute proposal. Later GMD (German National Research in Computer Science), was merged into Fraunhofer. GMD was more research oriented than Fraunhofer. Fraunhofer is really applied or contract research and GMD was more like basic and applied research. It did not as much research as the Max Plank Society. For example. Max Plank doesn't have to go out for projects. They have enough funding to perform their research; consequently they do more basic research. GMD was an in-between organization. At that time there was an outside research institute which actually fell apart working in the documentation field. They were chartered to develop things for online databases, like CAS (Chemical Abstract Services) in the USA, or FIS (Technical Information Systems) in Germany, but were not successful. In the end they were dissolved. A number of people left, and GMD was supposed to take over the remaining parts, and thus needed a leader for that. That's how they approached me. It was a challenge to build up from 20 people to the kind of what was later an institute of 100, 120 and more people, more than many American university CS departments.

So did the whole thing become contract research?

No, it was part of it. I mean, at GMD, my institute had about 50% that was directly government funded, and another 40% was typically EU funded with joint projects the EU was offering, and some from the German National Science Foundation. But this was a smaller part. Industry played a small role: a maximum 5-10% of the funding came from them. At Fraunhofer, this changed. They actually expect you to have about 30-40% of industry money, and then another let's say 30-40% of EU money, and only 20% would be given to you, directly. It is a very different model, especially as the pressure is on getting industry money. If you are not successful with the industry money, you are losing basic money also. You are in a situation where failure here will actually lead to your institute being out of money even if successful in other areas.

Well, how do you go after industry money? I think a lot of our readers might be interested in how you get money out of industry.

It is a lot of footwork. We went to fairs and exhibitions with demonstrations and showed them our capabilities. I'll give you the attitude of, let's say, a university person, and the attitude of somebody who wants industry money. My people came back from a fair and happily told me "we distributed all our business cards. Everybody was interested". And my question was, "how many business cards did you collect"? And they would just look at me, and say "huh"? Of course, as any industry person will know, you have to collect the business cards, because you have to make calls after that, and then you say "Yes, you were at my stand, and you saw my demonstrations. Are you interested in talking to us? We can probably help you". In Germany this is difficult, but also in Europe altogether. I believe that happens in computer science, not in machinery or manufacturing or other fields because there is very little primary industry in CS, nothing like Microsoft, IBM, and all those. What we have are application industries, banks and logistic companies and other similar ones. They don't want to take your prototype because people want product quality software. Essentially, they want your consulting, but you need money for doing research, and this is a stretch, it is not easy to bridge. You have to really learn how to do that. We had training. We had people that came in to teach us when GMD was taken over by Fraunhofer. One of them was a friend of mine. He headed IT for a bank in Germany. He would come and sit there and my people would come and make their 'sales' presentation. If he didn't like it he would say "thank you very much for an excellent presentation, don't call me, I will call you". This of course was a clear indication he wasn't impressed. But then he also analyzed the talk and told them why he wasn't impressed.

Even in the era of the web, the computer science is not topmost in the minds of the people.

Oh, that would be very helpful! Why is it so hard to get the scientific community to accept computer science as a first class discipline?

Because they are used to use computers as a service. They think computers are there, and computer science is just the industry that builds computers, but of course it is not. Very often, in research environments, the researchers write software themselves. Take a physicist, for example, in CERN, or in Stanford. Most of the software written there is not written by computer science people. It is written by physicists. So they feel that computer science is a service discipline for them, and not a standing by itself research field.

So do you agree with them, are we a service industry?

Of course I do not agree, but we had the same problem in Fraunhofer. Fraunhofer spreads to all disciplines and the largest institutes, the very successful institutes are in, let's call it mechanical

engineering. They work with the Germany car manufacturers, and they work with the famous German tool manufacturers and so on. Fraunhofer was created after the war to actually help build up the German industry again. And you know the success story of Germany is in the manufacturing industry. So these institutes considered that we, the IT institutes, would be essentially a service for them. They essentially came to us and told us "can you adjust this for us, or can you do this for us?" They were not interested in a partnership. They looked at us as a service. The Fraunhofer directors have a meeting once a year, where all directors are together for two or three days, and usually we have discussions about that view, because we IT people of course would not accept such a secondary role.

Did they ever change their mind, or is there something we need to do in our discipline to change their minds for them?

I think it is difficult. I think even in the era of the Web, the computer science is not topmost in the minds of people. If you go to Facebook and play around, or if you go to YouTube, or Twitter, or whatever, you are not thinking that behind all this is a computer scientist. You think of it as if someone just writes software that you can use, but seeing it as a discipline that analyses the data, for the good or for the bad (I don't want to comment on that), and a whole scientific research field behind it, I don't think is in front of the mind of the people. Maybe it needs time. I think mathematics may be in a very similar situation. However mathematics is accepted as a separate discipline, but they have a 3,000 years old history, and we have 40 years of history.

What do you think will happen with the conference system way of publishing in computer science?

I think we are in a time of changes. Because of the internet and the electronic availability of documents, now proceedings and the conference documents appear in a digital library essentially at the same time as the conference takes place, so the need to access documents by going to a conference will go away. Before, it was precisely that need, especially in the beginning, when we didn't have many journals. The way of publishing high quality work was through a conference. This was the main role, and that is changing now. The conference becomes more and more a social gathering place, because you can take the proceedings home, or you can download them later, but you are meeting peers, you go to sessions to discuss. All the conferences have now a tendency, like SIGMOD this year, to have the papers shortened in order to have more interactive sessions. There are people who have a paper and also have a presentation with a poster. We don't call it poster sessions now, but it is an additional possibility to discuss with the authors directly for better interactions. This social gathering aspect is increasing. But of course the other aspect, namely to have an achievable document in the database conference instead of a database journal will decrease because of that. At least, that is my opinion. But there is another situation coming up. I believe that maybe for the next generation of researchers (the one that is now 10 years old), physical gatherings may be in danger. This is because they have all this Facebook stuff. I see it with my grandchildren. Funny as it is, they are sitting in the same room, and talk to each other via Facebook, and not directly. They have this group of Facebook friends, and they chat and chat, and exchange all kinds of information. When they meet the same people physically, they don't know what to talk about. In a way, I believe there is a danger that the conferences as a social gathering place will also disappear, but then what is remaining and what are we loosing?

Definitely having a drink or a coffee in a bar together. I am watching this development very carefully, but I see a tendency in this direction and I believe it is wrong.

There is a danger that the conferences as a social gathering place will also be gone.

Do you have any words of advice for fledgling or midcareer database researchers or practitioners?

My advice is that you look around. See "hanging in the air" problems. I mean, at the conferences, sometimes good keynotes will raise issues. Look around your own environment and see some problems, and then you try to formulate a goal. What would you like to see happening here? And then try to find an approach of how you can solve it or partially solve it, if it is a mega problem. I always keep saying to my PhD students, "you should also be willing to say no to your advisor", because he is caught in his own context, and he may not see some developments. I changed fields a number of times. I worked not only on the database field, but earlier in programming languages and later also the in the digital library field. But when you are older, it is very hard to establish yourself in a new field. So for that reason, you get kind of stuck in your field. But young people or midcareer people are still free. They can still make this change. So that is my advice: follow your own intuition and not necessarily your advisor or a prospective (short term) job situation.

If you magically had enough extra time at work to do one thing that you are not doing now, what would it be?

I would really go and look at the whole issue of interoperability and semantics and modeling and try to make some sense out of the many conflicting and many repeating approaches. I'd try it for myself. First of all, find out whether there was any progress in the last 20 years in the field. But then also try to find out what would be a good thing to really help a human centered approach, to really help the human to find things she/he does not know at all. Most of the approaches in semantics, even if they are semi-automatic and with human feedback and all such stuff, always assume the human knows. But if I don't know, I can make no judgment. If I go to, let's say, a medical wiki and I believe I have this and that sickness, and I find treatments in there, how do I know I can trust this information? I mean, I can only do it when I know doctors that have a high reputation or experience, and they back these descriptions. But I don't know the doctors because the whole subject is new to me. My human feedback is not going to help either. The question therefore is: how can you build systems that do not assume the human knows what he/she is doing?

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

I don't know. I started in electronics, I have to say, and I switched to computer science only when I joined IBM. I became a programmer, and the group itself was theoretically oriented,

consequently I did formal language stuff for programming. But when I got in contact with databases, I realized that one thing that would have helped me a lot, even in the early days with SQL and relational approaches would be to know more about linguistics. I feel knowledge about linguistics, concepts of speech, and natural languages, etc., is very helpful. That is even more helpful and more important nowadays. When I started IPSI, I had a linguistics group as one of five research fields. But we needed additional research money and eventually I couldn't maintain the group. I would get projects, but I would not get linguistics projects because 'true' linguists at the university would get that part, even of cooperation projects. Here I regret that I had no linguistic background myself. I was an outsider, and the funding agencies would not give me the money. They gave me the implementation aspects, the software aspects of the project, but not the linguistic aspects. If I had been a linguistic, even as a second kind of expertise I probably would have been more successful.

Thank you very much for talking with me today.

Thank you very much for having me.

A Call for Surveys

Philip A. Bernstein
Microsoft Corporation
philbe@microsoft.com

Christian S. Jensen
Aarhus University
csj@cs.au.dk

Kian-Lee Tan
National Univ. of Singapore
tankl@comp.nus.edu.sg

The database field is experiencing an increasing need for survey papers. We call on more researchers to set aside time for this important writing activity.

The database field is growing in population, scope of topics covered, and the number of papers published. Each year, thousands of new papers enter the database research literature. As a result, it has become a daunting task to maintain a basic understanding of more than a few major areas of database technology. Even relatively narrow topics have dozens of papers, making it hard for students, researchers, and engineers to get a quick overview of the state-of-the-art.

The increasing demand for surveys has recently been recognized by commercial publishers, such as Morgan-Claypool with their Synthesis Lectures series, Now Publishers with their Foundations and Trends in Databases series, and Springer with their SpringerBriefs series. There is also a Surveys section of SIGMOD Record. We applaud these efforts, but feel that the field would benefit from a great many more surveys than are currently being published.

While the raison d'être of research journals is primarily to publish original research results, many journals welcome the submission of surveys. However, few submissions are received. In particular, The VLDB Journal has always welcomed surveys, but the submission rate is lower than we would like. To further encourage the submission of surveys, we offer prospective authors of a survey the option of contacting us in order to gauge the level of interest before investing the effort.

In addition to the altruistic reason of writing a survey to help other researchers, there are also selfish reasons to invest time to write a survey. A survey is great way for junior and senior researchers alike to establish a presence in a research area and to become better known to a wider community. It is also a good way to increase their citation count. For example, according to Microsoft Academic Search, the two most highly-cited papers in The VLDB Journal are surveys. And according to Springer, a third survey is the most frequently downloaded paper from the journal.

New Ph.D. graduates who worked on a well developed topic have already done a literature review which, possibly in collaboration of their advisor, could be extended into a high-quality survey paper. Similarly, researchers who have prepared tutorials at research conferences are also well on their way to generating an excellent survey on a timely topic.

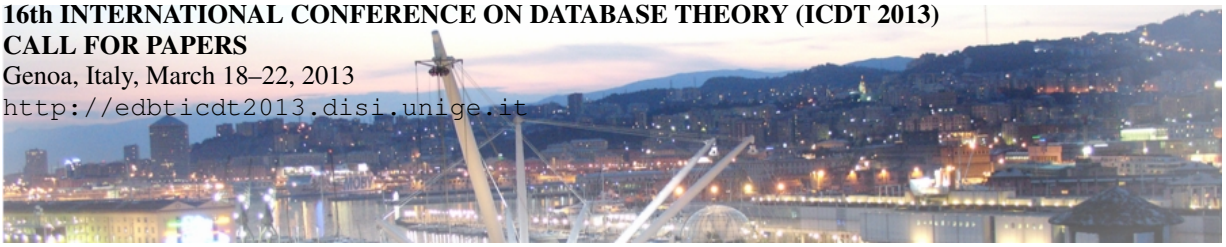
A survey covers an area with a well-established body of work rather than introducing novel solutions in the surveyed area. Successful surveys are authoritative and offer comprehensive coverage within their chosen scope. They may offer broad coverage of an area or in-depth coverage of a narrower subarea. A survey can add value by synthesizing new ways of understanding the relationships among previous contributions. In addition to reviewing the key results in an area, a survey may give the reader a deep understanding of goals, requirements, solutions, open questions, and example systems and applications. A survey that analyzes, integrates, and classifies what is known about a topic in a clear and comprehensive fashion by means of a conceptual framework is a tool for thought that provides a baseline from which the field can make faster progress, and it represents a valuable contribution in its own right.

The VLDB Journal has no minimum length for surveys, but they must comply with the general maximum length restriction for papers.

About the Authors: The authors are currently the editors-in-chief of The VLDB Journal, published by Springer.

See `http://vldb.org/vldb_journal`.

16th INTERNATIONAL CONFERENCE ON DATABASE THEORY (ICDT 2013)
CALL FOR PAPERS
Genoa, Italy, March 18–22, 2013
http://edbticdt2013.disi.unige.it



Invited Speakers
Jan van den Bussche (Hasselt Univ.)
Yehoshua Sagiv (Hebrew Univ. of Jerusalem)
Luc Segoufin (Inria & ENS-Cachan)

Program Chair
Wang-Chiew Tan
(IBM Almaden and UC Santa Cruz)

Publicity Chair
Benny Kimelfeld (IBM Almaden)

Program Committee
Balder ten Cate (UC Santa Cruz)
James Cheney (University of Edinburgh)
Jan Chomicki (University at Buffalo)
Sara Cohen (The Hebrew University of Jerusalem)
Todd J. Green (UC Davis & LogicBlox)
Sudipto Guha (University of Pennsylvania)
Benny Kimelfeld (IBM Almaden)
Solmaz Kolahi (Oracle)
Kobbi Nissim (Ben-Gurion University)
Antonella Poggi (Sapienza University of Rome)
Riccardo Rosati (Sapienza University of Rome)
Cristina Sirangelo (ENS Cachan)
Nicole Schweikardt (Goethe-University Frankfurt)
Kyusheok Shim (Seoul National University)
Sławek Staworko (University of Lille)
Jianwen Su (UC Santa Barbara)
Wang-Chiew Tan (IBM Almaden & UC Santa Cruz)
Stijn Vansummeren (University Libre de Bruxelles)
Victor Vianu (UC San Diego)
Jef Wijsen (University of Mons)

Important Dates
Abstract submission deadline:
August 17, 2012, 11:59pm PDT (firm)
Paper submission deadline:
August 24, 2012, 11:59pm PDT (firm)
Notification deadline: *Nov. 2, 2012*
Camera-ready deadline: *TBD*
Conference: *March 18–22, 2013*

The series of ICDT conferences provides an international forum for the communication of research advances on the principles of database systems. Originally biennial, the ICDT conference has been held annually and jointly with EDBT ("Extending Database Technology") since 2009.

Topics of interest for submissions include but are not limited to:

Business processes and workflows; Concurrency and recovery; Constraint databases; Data exchange and data integration; Data mining; Data models, semantics, and query languages; Data privacy and security; Data provenance; Data streams; Inconsistency and uncertainty in databases; Information extraction; Deductive databases; Distributed and parallel databases; Logic and databases; Query processing and optimization; Semi-structured and Web data; Spatial and temporal databases; Transaction management; Views and data warehousing.

Submissions: Papers must be submitted electronically to: https://www.easychair.org/conferences/?conf=icdt2013.

Papers must be submitted as PDF documents within 12 pages according to the ACM guidelines, which can be found at http://www.acm.org/sigs/publications/proceedings-templates. Templates are available in Word, WordPerfect, and LaTeX (versions 2.09 and 2e). For the LaTeX formats, you may use either the standard style or the SIG-alternate style. It is not permissible to change the template's font size, margins, inter-column spacing, or line spacing, etc. If the authors believe more details are necessary to substantiate the main claims of the paper, they may also include a clearly marked appendix to be read at the discretion of the committee. Papers not conforming to these requirements may be rejected without further consideration.

The deadline for abstract submission is August 17, 2012 and the deadline for paper submission is August 24, 2012. Submission deadlines firm; late submissions will not be considered. Authors will be notified of acceptance or rejection by November 2, 2012.

The results must be unpublished and not submitted for publication elsewhere, including the formal proceedings of other symposia or workshops. All authors of accepted papers will be expected to sign copyright release forms. One author of each accepted paper will be expected to present it at the conference. The proceedings will appear in the ACM International Conference Proceedings Series (pending approval).

Awards: An award will be given to the best paper. Also, an award will be given to the best paper written by newcomers to the field of database theory. The latter award will preferentially be given to a paper written only by students; in that case the award will be called "Best Student Paper Award." The program committee reserves the following rights: not to give an award; to split an award among several papers; and to define the notion of a newcomer. Papers authored or co-authored by Program Committee members are not eligible for any award.