The Association for Computing Machinery, Inc.
1515 Broadway
New York, NY 10036
U.S.A.

ACM ISBN: 1-59593-160-0

Printed by AT&T Labs-Research, USA.

Preface

The problem of poor data quality stored in database-backed information systems is widespread in the governmental, commercial and industrial environments. Alarming situations with various information quality problems can not be ignored anymore and theoretical as well as pragmatic approaches are urgently needed to be proposed and validated. As a consequence, information quality is now becoming one of the hot topics of emerging interest in the academic and industrial communities.

Many processes and applications (such as information system integration, information retrieval, and knowledge discovery from databases) require various forms of data preparation or repair with several data processing techniques, because the data input to the application-dedicated algorithms is assumed to conform to nice data distributions, containing no missing, inconsistent or incorrect values. This leaves a large gap between the available "dirty" data and the available machinery to process the data for application purposes.

The Second Edition of the International Workshop IQIS 2005 (Information Quality in Information Systems) is held in Baltimore, MD, USA, on June 17, 2005. The workshop is sponsored by ACM and in conjunction with the Symposium on Principles of Database System (PODS) and the ACM SIGMOD International Conference on Management of Data. IQIS workshop focuses on database-centric issues in data quality (scalability, quality-aware query processing, applications like data integration). It intends to address methods, techniques of massive data processing and analysis, methodologies, new algorithmic approaches or frameworks for designing data quality metrics in order to understand and to explore data quality, to find data glitches (as data quality problems such as duplicates, errors, outliers, contradictions, inconsistencies, etc.) and to ensure both data and information quality of database-backed information systems.

The program and the organization of the workshop are the result of a huge effort by many people who contributed to IQIS 2005 and we want to warmly thank them all. First, we would like to thank the authors of all submitted papers, both accepted and rejected ones.

The 11 papers collected in this volume, out of 26 papers that were submitted (with 10 short papers and 16 research papers), are a significant sample of recent achievements in the various areas of information and data quality, ranging from quality models to record linkage and statistical techniques.

Then, we wish to thank the program committee members for the reviewing work they did to ensure high quality papers.

We wish to thank our distinguished keynote speakers Hector Garcia-Molina, from Stanford University, USA, who described entity resolution process and algorithms, and William E. Winkler, from U.S. Bureau of the Census, USA, who described methods and techniques for massive data clean-up.

We would like also to warmly thank Felix Naumann, from Humboldt-Universität zu Berlin, Germany, and Monica Scannapieco, from Università di Roma "La Sapienza", Italy, for the organization of the very first edition of the IQIS workshop, last year in Paris, in conjunction with ACM SIGMOD/PODS 2004 conference. Without their initiative, enthousiasm and motivation, the second edition of IQIS would not exist.

In addition, we would like to thank all the people who volunteered their time to help us organize the workshop. In particular, we would like to thank Laurent Amsaleg for his precious recommendations, Élisabeth Lebret, Philippe Lecler and, of course, Marianne Winslett and Lisa Singh for taking care of overall and local workshop organization and all the related issues.

Finally, we thank you for attending the IQIS 2005 workshop. We sincerely hope that you find the program very exciting and enjoy the workshop environment.

Laure Berti-Equille
Carlo Batini
Divesh Srivastava
IQIS 2005 Workshop co-chairs

Program Committee

Boualem Benattallah, *Queensland University of Technology, Australia*

Mokrane Bouzeghoub, *Université de Versailles, France*

Tiziana Catarci, *Università di Roma "La Sapienza", Italy*

Tamraparni Dasu, *AT&T Labs-Research, USA*

Johann-Christoph Freytag, *Humboldt-Universität zu Berlin, Germany*

Helena Galhardas, *INESC-Lisboa, Portugal*

Michael Gertz, *University of California, USA*

Ahmed K. Elmagarmid, *Purdue University, USA*

Markus Helfert, *Dublin City University, Ireland*

Matthias Jarke, *RWTH Aachen, Germany*

Theodore Johnson, *AT&T Labs-Research, USA*

Vipul Kashyap, *National Center for Biotechnology Information, USA*

Yang Lee,*Northeastern University, USA*

Bing Liu, *University of Illinois at Chicago, USA*

Paolo Missier, *University of Manchester, United Kingdom*

Tamer Ozsu, *University of Waterloo, Canada*

Ronald Pearson, *ProSanos Corporation, USA*

Leo Pipino, *University of Massachusetts Lowell, USA*

Mario Piattini, *University of Castilla - La Mancha, Spain*

Louiqa Raschid, *University of Maryland, USA*

Giri Kumar Tayi, *Albany University, USA*

Panos Vassiliadis, *University of Ioannina, Greece*

Richard Wang, *Massachussetts Intitute of Technology, Boston, USA*

Table of Contents

IQIS 2005 Keynote Speech 1
Handling Data Quality in Entity Resolution

Hector Garcia-Molina

Stanford University
California, USA

ABSTRACT

Entity resolution (ER) is a problem that arises in many information integration scenarios: We have two or more sources containing records on the same set of real-world entities (e.g., customers).

However, there are no unique identifiers that tell us what records from one source correspond to those in the other sources.

Furthermore, the records representing the same entity may have differing information, e.g., one record may have the address misspelled, another record may be missing some fields.

An ER algorithm attempts to identify the matching records from multiple sources (i.e., those corresponding to the same real-world entity), and merges the matching records as best it can.

In many ER applications the input data has data quality or uncertainty values associated with it. Furthermore, the ER process itself introduces additional uncertainties, e.g., we may only be 90% confident that two given records actually correspond to the same real-world entity.

In this talk Hector Garcia-Molina will discuss the challenges in representing quality/uncertainty/confidences in a way that is useful for the ER process.

He will also present some preliminary ideas on how to perform ER with uncertain data. (This work is joint with Omar Benjelloun, David Menestrina, Qi Su, and Jennifer Widom).

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

ABOUT HECTOR GARCIA-MOLINA

Hector Garcia-Molina is the Leonard Bosack and Sandra Lerner Professor in the Departments of Computer Science and Electrical Engineering at Stanford University, Stanford, California. He was the chairman of the Computer Science Department from January 2001 to December 2004. From 1997 to 2001 he was a member the President's Information Technology Advisory Committee (PITAC).

From August 1994 to December 1997 he was the Director of the Computer Systems Laboratory at Stanford. From 1979 to 1991 he was on the faculty of the Computer Science Department at Princeton University, Princeton, New Jersey. His research interests include distributed computing systems, digital libraries and database systems.

He received a BS in electrical engineering from the Instituto Tecnologico de Monterrey, Mexico, in 1974. From Stanford University, Stanford, California, he received in 1975 a MS in electrical engineering and a PhD in computer science in 1979.

Garcia-Molina is a Fellow of the Association for Computing Machinery and of the American Academy of Arts and Sciences; is a member of the National Academy of Engineering; received the 1999 ACM SIGMOD Innovations Award; is a member of the Computer Science and Telecommunications Board (National Research Council); is on the Technical Advisory Board of DoCoMo Labs USA, Kintera, Metreo Markets, TimesTen, Verity, Yahoo Search Marketplace; is a Venture Advisor for Diamondhead Ventures, and is a member of the Board of Directors of Oracle and Kintera.

IQIS 2005 Keynote Speech 2
Methods and Analyses for Determining Quality

William E. Winkler

U.S. Bureau of the Census,
Statistical Research
USA

ABSTRACT

In a possibly ideal world, records in a database would be complete and would contain fields having values that correspond to an underlying reality. An individuals name, address and date-of-birth would be present without typographical error. An income field might be a reasonably close approximation of a "true income" and would not be missing. A list of customers would be complete, unduplicated and current.

In this ideal world, a database could be used for several purposes and would be considered to have high quality. A set of databases might be linked using name, address, and other weakly identifying information.

In this paper, we describe situations where properly chosen metrics may indicate that data quality is not sufficiently high for monitoring processes, for modeling, and for data mining.

Some of the metrics are supplementary to those in the quality literature or have rarely been used. Additionally, we describe generalized methods and software tools that allow a skilled individual to perform massive clean-up of files in some situations.

The clean-up, while possibly sub-optimal in recreating "truth", can replace exceptionally large amounts of clerical review and allow many uses of the "cleaned" files.

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

ABOUT WILLIAM E. WINKLER

Ph.D. Probability Theory
Principal Researcher, US Census Bureau
Fellow, American Statistical Association

Provider issues in quality-constrained data provisioning

Paolo Missier and Suzanne Embury
School of Computer Science
The University of Manchester, UK
{s.embury,pmissier}@cs.manchester.ac.uk

ABSTRACT

Formal frameworks exist that allow service providers and users to negotiate the quality of a service. While these agreements usually include non-functional service properties, the quality of the information offered by a provider is neglected. Yet, in important application scenarios, notably in those based on the Service-Oriented computing paradigm, the outcome of complex workflows is directly affected by the quality of the data involved. In this paper, we propose a model for formal data quality agreements between data providers and data consumers, and analyze its feasibility by showing how a provider may take data quality constraints into account as part of its data provisioning process. Our analysis of the technical issues involved suggests that this is a complex problem in general, although satisfactory algorithmic and architectural solutions can be found under certain assumptions. To support this claim, we describe an algorithm for dealing with constraints on the completeness of a query result with respect to a reference data source, and outline an initial provider architecture for managing more general data quality constraints.

1. INTRODUCTION

An increasing number of information providers nowadays offer query services on large data sets through internet-wide published interfaces, using a variety of widely available technologies. Alongside the definition of a service interface, the stipulation of agreements regarding the quality of the service is also becoming commonplace, eg. in the form of *Service Level Agreements* [12, 2, 20]. Such agreements, however, only deal with performance issues, while the quality of the information delivered to service users is generally neglected. When compared to the more common experience of shopping for any kind of product, this situation is akin to assuming that the customers' only issue is with the opening hours of the store or the service time at checkout, with no concern for the quality of the goods – clearly an unrealistic expectation.

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

We argue that data consumers are in a similar predicament: the sizable and mature body of knowledge regarding quality properties of data [18, 19, 24, 21, 9] does not translate into actionable user requirements, and yet, in simple and realistic scenarios, specific properties of data are important. Suppose, for instance, that a provider acquires copyrighted articles from publishers, and compiles independent digests and reviews of those articles, offering them for sale. While users are interested in purchasing the added-value reviews, they also want to make sure that by doing so, they are not missing the digest for any of the articles that would meet their criteria if requested directly to the publishers. For example, they want to purchase the digest for the ten most recent papers on a particular topic.

The idea at the core of our work is that users may enforce this and similar requirements by entering into a formal agreement with the added-value provider, whereby the digests produced in response to a query are guaranteed to include a sufficiently large fraction of the articles that would have been returned, had the same query been issued directly to the publisher. We refer to this property of the data as its *completeness* relative to a reference data source –in this case, the original publisher.

Thus, a completeness constraint is intended to differentiate between providers that only offer digests for a small subset of the articles that are actually available, and those that account for larger collections. Notice that, in this example, the quality of the digest itself is not part of the agreement, although it may be similarly formalized as a quality constraint, of a different type: completeness is only one of many possible properties of data for which constraints can be defined.

This simple scenario is becoming increasingly relevant in situations where (i) the data obtained from a provider has a quantifiable value to the consumer, (ii) its worthiness depends on one or more quality properties, and (ii) multiple providers may offer similar information. The combination of these factors contributes to the development of a data marketplace, whereby consumers that are interested in quality data negotiate its quality/cost trade-offs with providers. The value of quality as perceived by consumers is not necessarily only monetary. Consider for instance the case, also increasingly important, of a biologist who performs data-intensive experiments using various algorithms that operate on data obtained from public repositories (so-called *in silico* experiments). For example, the success of a gene sequence similarity analysis, consisting of matching a string sequence against a large database of known sequences, depends on

the completeness of the reference data source. Although the experiment's success criteria are normally not expressed in monetary terms, the value of using a complete reference data set is unquestionable.

The missing element that would enable data marketplaces is the ability for data providers and consumers to negotiate formal and binding agreements regarding the quality of the data. In this respect, providers seem to face the greatest challenges, as they must determine which agreement levels can be sustained, and the cost/benefit trade-offs involved. To the best of our knowledge, these issues have not been addressed, with the exception of a 1989 paper by Ballou and Tayi [5], discussed later.

This paper attempts to fill this gap. Its core contribution is a model for quality agreements, and an analysis of the issues and possible strategies available to providers that commit to such agreements. We begin by assuming that every data transaction, consisting of a query-result pair, may be subject to quality constraints. Before any such transaction may occur, providers and consumers should agree on definitions for the following elements:

- a pricing function, which associates a value to the result of any query issued by the consumer;
- a number of quality functions that formalize the notions of quality properties, and that associate a quality value to each result;
- a function of quality values for the result, that quantifies their appropriateness to the consumer. This function, not necessarily linear, is expressed as a *penalty* whose effect is to reduce the price paid for the result.

The negotiation process that leads to the specific definition of each of these elements is not relevant for our purposes, and is not considered in this paper. Before entering into such an agreement, the provider must determine its feasibility, by assessing (i) the actions required to provide data with the required quality values, and (ii) whether its data architecture supports those actions in a cost-effective way. Specifically, for each incoming query, the provider faces two problems:

1. **Detection:** it must determine to what extent a result set would incur any penalty, due to insufficient quality levels, for all the quality properties involved;
2. **Repair:** it must determine what actions are available to repair its data in order to reduce or avoid the penalties.

We base our model on the assumption that both detection and repair have a cost, forcing the provider to solve an optimisation problem involving penalties, price, and costs.

As is common with any marketing scenarios, the provider may adopt a number of different strategies for compliance. For instance, it may invest resources to proactively ensure that most of its data comply with the constraints, regardless of the specific user requests. More realistically, it may conservatively adjust the quality levels of its data, by observing the consumer's behaviour – for instance, by investing in quality only for the most popular data and accepting penalties for less frequently requested items.

Rather than focusing on any specific model, in this paper we define the provider's problem space by enumerating the factors that affect its strategies. This results in a general framework that can be used to analyze complex scenarios. The most critical factor concerns the amount of information available to ensure that the penalties assessed are *provably fair*: if we assume that the actual value of a quality function is always available both to the provider and the consumer, then (i) the fairness of the agreement can be verified, and (ii) the provider may implement an optimal provisioning strategy. For some quality properties, however, this assumption may not be realistic. For instance, evaluating the completeness of a data set relative to a reference set requires full knowledge of the latter. We model the problem of partial knowledge of quality by attaching a cost to the evaluation of quality functions; in the case of completeness, this would be the per-item cost of querying the reference source, in order to obtain a partial view of its contents. This leads to the formulation of quality estimates, which put the fairness of the agreement into question, and compromise the optimality of the provider's strategy. We propose (Section 3.1) to deal with fairness issues by introducing a third-party verification role and using a spot-check approach for ensuring limited but acceptable fairness.

As additional contribution, we present an algorithm for dealing with the specific case of data completeness constraints, first using the most favorable assumptions, including availability of quality values, and then in the more general case of partial availability. This provides an insight into the expected complexity of the general provisioning problem, and also shows a case of a property-specific algorithm that cannot be easily reused for other properties; it suggests that the generality of the solution may be limited to the detection-maintenance pattern.

As a final contribution, we describe (Section 6) a general data provider architecture that incorporates that pattern, and show how it can be implemented alongside a standard query processing engine.

Our initial investigation into the problem of provisioning data with quality constraints shows that this is a difficult one in general, although satisfactory algorithmic solutions can be found under realistic assumptions.
In the rest of the paper, quality functions are introduced in Section 2, followed by the agreement model in Section 3. The provider model is presented in Section 4, and the specific handling for completeness in Section 5. The reference architecture is discussed in Section 6. We conclude in Section 7 with our agenda for further work.

1.1 Related work

Although the specific topic of data quality agreements is new, some authors address related problems, specifically in the area of quality-aware data integration. Naumann et al. [17] assume that scores can be assigned to the data offered by multiple providers, to reflect its quality, and show that the problem of data integration in the presence of such scores results in significant extensions to known algorithms for querying data using views. We tackle a somewhat complementary problem, namely how a provider can manage its data assets in order to *obtain desirable quality scores*, which would then be passed up to an integration mediator. Similarly, a recent paper by Motro [3] shows how using utility functions may alleviate the problem of data fusion (i.e., combining different versions of the same date) in the presence of inconsistencies. Utility functions are based on quality features such

as recentness and accuracy. Using a similar perspective, the "Quality Broker" architecture presented in [15] assumes that quality features are available from several sources. The goal of the architecture, in this case, is not quality constraint satisfaction, but rather the broker-based selection of the most appropriate answer to a query, among multiple available.

A related problem is also addressed in [4], where the notion of a *data quality certificate* is presented. The purpose of the certificate is to enable reasoning about quality within the context of cooperative information systems, in order to improve the overall quality of inter-system workflows. This notion is also used, in a different form, in the quality profile model described in [13].

Underlying all of these approaches are assumptions regarding (i) a shared underlying model for quality description, and (ii) the way quality values are actually computed. While none of them seems concerned with their actual availability, in some cases, the granularity of the quality meta data is so fine – at the level of a single attribute, that the actual feasibility of computing the corresponding values may be questioned. An important but overlooked problem then becomes, to assess the robustness of these integration processes when some of the quality data is missing.

Some authors define completeness by taking into account both the size of a data source and the number of available attributes with respect to the reference source, as well as the fraction of attribute values that are non-null. Our definition of completeness only considers the size of the relation, and not the single attributes, and thus it is simpler than the *relational completeness* found in [16]. It corresponds roughly to the notion of *coverage* introduced in [11]; there, coverage is defined with respect to a universal relation, whose extension includes all tuples obtained from a number of primary data sources. Our single reference source corresponds to the universal relation.

Ballou et al. [5, 6] presented an interesting and very pertinent early attempt at linking quality properties to the effort required to provision them. There, the goal is to determine, using an integer programming model, the most effective distribution of resources that a provider can use to maintain or enhance data integrity, each with an associated cost and effectiveness. While initially assuming precise knowledge of the underlying cost, data error rate and other parameters, the model also addresses the problem of estimating some of those parameters using heuristics. How realistic the overall model is in practice, however, remains to be seen.

Finally, following the intuition that information is but another type of product, some authors have adapted established results from the practice of quality control in product manufacturing, resulting in the IP-MAP (Information Product Map) framework [23, 22, 7]. For our purposes, the merit of this work is to provide ways to *predict* quality values based on the analysis of the processes that produce those values. However, we believe that the barebone model for completeness presented in the next section would defeat the purpose of such machinery, which is best suited for complex processes with well-identified "quality weak spots."

2. QUALITY FUNCTIONS

We begin by providing functional definitions of quality properties, which we illustrate for the case of completeness, defined earlier, and of consistency, i.e., the property of a data set to conform to some validation rule.

We only consider relational data sets, i.e., extensions of a relational schema. Given two data sets D and D_r, we define the completeness of D relative to D_r as:

$$compl(D, D_r) = \frac{|D \cap D_r|}{|D_r|} \in [0, 1] \quad (1)$$

In particular, we are interested in the completeness of the result $Q(D)$ of a query:

$$compl_Q(D, D_r) = \frac{|Q(D) \cap Q(D_r)|}{|Q(D_r)|} \in [0, 1] \quad (2)$$

Intuitively, $compl_Q()$ counts the fraction of records from D_r that the user obtains by querying D rather than D_r. Recall that the reason for querying D in our examples is that it contains *added value* versions of data from D_r.

This definition is illustrated in Figure 1. Note that $compl(D, D_r)$ may be very different from $compl_Q(D, D_r)$ for some Q; even when D contains only a small subset of D_r, its completeness relative to a particular query may be high, as long as D contains most of the items that the user is requesting. In fact, the heuristics for providing completeness, presented later, are based on the provider's knowledge of the user queries; their effectiveness depends on the predictability of those queries.

Before we proceed, we must first give a precise meaning to the expression $Q(D_r)$, by clarifying the relationship between D and D_r. Let S_D and S_{D_r} be the relational schemas for D and D_r, respectively. Following the well-known "Global-as-View" pattern, described for instance by Lenzerini [14], we assume that S_D is defined as a view on S_{D_r}. Formally, this requires the definition of a mapping query mq over S_{D_r}, written as $S_D \rightsquigarrow mq(S_{D_r})$, so that any query issued to S_D can be translated into a corresponding query to S_{D_r}, through a simple process of *unfolding* of the mapping query. For example, suppose that S_{D_r} consists of two relations, $R_1(a_1, a_2, a_3)$ and $R_2(b_1, b_2, b_3)$, and that S_D consists of relation R, defined by the mapping query:

$$R(c_1, c_2, c_3) \rightsquigarrow \pi_{a_1,b_2,b_3}(\sigma_{a_3=c}(R_1 \bowtie_{a_2=b_2} R_2))$$

Then, for a query like

$$Q \equiv \pi_{c_1}(\sigma_{c_2=x}(R)),$$

the corresponding query on S_{D_r} used in the completeness definition would be

$$Q' \equiv \pi_{a_1}(\sigma_{b_2=x \wedge a_3=c}(R_1 \bowtie_{a_2=b_2} R_2))$$

To simplify the notation, in the rest of this paper we write $Q(D_r)$ instead of $Q'(D_r)$.
It is also worth mentioning that $Q(D) \cap Q(D_r) = Q(D) \cap D_r$: the use of $Q(D_r)$ only really matters in the denominator of expression (2).

As a second example of functional definition of quality property, we also define the consistency of a data item relative to a conformance rule; for instance, a rule may state that a street address in a database entry is consistent with a reference street atlas, if it can be matched uniquely against one of the reference streets in the atlas. The record matching problem has been studied extensively in the data quality literature [10, 8, 1] [1], and it is known that the evaluation of

[1]W.Winkler has made a rich collection of references to this problem available at http://csaa.byu.edu/kdd03-papers/winkler-refs.pdf.

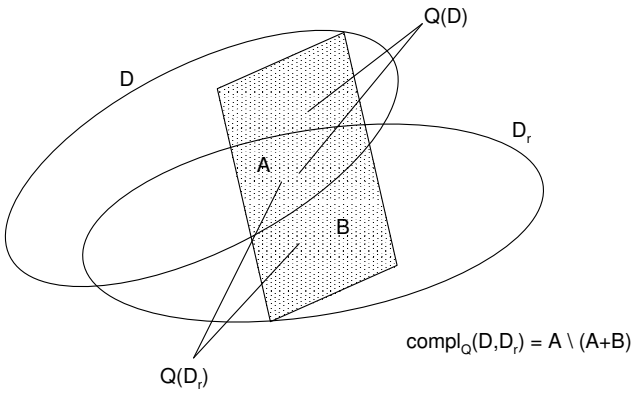


Figure 1: Completeness of a query result

this rule may incur uncertainty, accounting for the chance of false positives (an erroneus match). Thus, we assume that the rule is a function of the data and of some parameters (i.e., the reference source), and that its evaluation produces a "yes/no" result, along with a level of confidence. This function does not perform any correction or issue recommendations. In this regard, it behaves like an integrity constraint checker on a database schema.

The validation function for an item $d \in D$ with respect to D_r is

$$val(d, D_r) \in (\{true, false\}, [0, 1])$$

where the second component of the value is the confidence in the outcome. Given a user-defined threshold c_0 for the confidence, let the set of *acceptable* items relative to *val* and c_0 be

$$acc(val, D, D_r, c_0) = \{d \in D | val(d, D_r) = (true, c) \wedge c \geq c_0)\}$$

A simple definition for the consistency of D relative to D_r, given c_0, is the proportion of acceptable items in D. For $Q(D)$ this is written as:

$$cons_Q(val, D, D_r, c_0) = \frac{|acc(val, Q(D), D_r, c_0)|}{|Q(D)|} \in [0, 1]$$

This simple definition illustrates the general idea of consistently defining normalized quality functions, using user-specified parameters such as the threshold c_0.

3. AGREEMENT MODEL

Agreements are based on a simple penalty/reward model, whereby the consumer and the provider agree on formally defined quality constraints for the data provisioned on a query-by-query basis, the provider may charge fees in return for data, and the consumer may assess penalties when the quality constraints are violated.

Rather than defining discrete constraints, i.e., of the form $compl_Q(D, D_r) > compl_{min}$, we instead allow for a more general formulation, by associating penalty functions of arbitrary shapes to quality functions. When applied to a base price for a query result, they reduce the actual fee paid, in a way that is proportional to the perceived importance of the specific property.

For query Q on D, the agreement includes the following elements:

- a set of normalized quality functions of the form

  $$qf(D', \mathcal{P}) \in [0, 1]$$

  for any $D' \subseteq D$, where 1 is the best quality achievable. $\mathcal{P}$ indicates a property-specific set of additional parameters, eg. D_r, c_0. In particular, we are interested in computing the quality associated to a query result, $qf(Q(D), \mathcal{P})$;

- a base pricing function $price_b(D')$ for any $D' \subseteq D$. Again, in practice we are interested in computing $price_b(Q(D))$;

- a penalty function $pen_{qf}(D', \mathcal{P}) \in [0, 1]$, which introduces a bias on the base price, by mapping the values of the normalized quality function qf applied to D', onto a penalty factor.

The actual price paid by the user for $Q(D)$ is then

$$price(Q(D), qf(), \mathcal{P}) = price_b(Q(D)) \times (1 - pen_{qf}(Q(D), \mathcal{P}))).$$

Note that discrete quality constraints can be expressed simply by defining binary penalty functions. For example, the following constraint makes the result set worthless if the completeness falls below a threshold $compl_{min}$:

$$pen_{compl}(D', D_r) = \begin{cases} 0 & \text{if } compl(D', D_r) > compl_{min} \\ 1 & \text{otherwise} \end{cases}$$

Normally, however, the penalty will be proportional to the quality level, i.e., the function is monotone decreasing in the value of $qf()$. Although no assumptions on the shape of penalty and pricing functions need to be made, this additional information helps in reducing the complexity of the provider algorithms for managing quality compliance. The baseline algorithm presented in Section 5.1, shows a worst case scenario, in which no assumption is made regarding the shapes of these functions.

Note also, that using normalized quality functions makes it straightforward to extend this pricing scheme to multiple quality properties, i.e., by combining multiple quality and penalty functions:

$$price(Q(D), \{qf_i()\}, \{\mathcal{P}_i\}) = price_b(Q(D)) \cdot \prod_i (1 - pen_{qf_i}(Q(D), \mathcal{P}_i)))$$

Functions $\{qf()_i\}$, $price_b()$ and $\{pen_{qf_i}()\}$ are all defined as part of a negotiation process, whose details are not relevant for our purposes. Once the agreement is in place, it is enforced as follows:

- for every incoming query Q, the provider computes $D' = Q(D)$;
- for every $qf_i()$ that is subject to a penalty, compute $q_i = qf_i(D', \mathcal{P}_i)$ and $p_i = pen_{qf_i}(q_i)$;
- compute the final price $price_b(D') \cdot \prod_i (1 - p_i)$.

3.1 Fairness of penalty assessment

As anticipated, the fairness of the penalty assessment depends upon the value of $qf()$ being available. In our example, this corresponds to having a catalog of all available articles, which can be queried (this is $Q(D_r)$), regardless of how many of those articles have received a review. Similarly, in the biology database case, a public source for the primary data may be available free of charge. While these are reasonable assumptions, in general we also need to consider the cost of computing $qf()$. When the provider incurs this *monitoring cost* (see Section 4), it may need to compute an estimator $\hat{qf}()$ of the actual $qf()$, balancing its precision with the cost of limited monitoring.
The idea is then to let providers self-assess their penalties, and to adopt the simple but widespread view that a third-party verification authority is in charge of assessing the correctness of penalties. We assume that this authority also incurs a monitoring cost. For completeness, this results in the following scenario:

- the provider defines its own estimators $\hat{qf}()$ for $qf()$, based on some probabilistic model and by sampling using a limited number of $Q(D_r)$ queries, determined by its budget[2];
- the authority has its own strategy for verification, which relies on spot-checks performed at some time intervals, again by querying $Q(D_r)$;
- the provider may negotiate a tolerance $\delta \in [0, 1]$ as part of the agreement, which limits its liability vs the customer in case of imprecise estimates;
- whenever the authority determines the actual value for $qf()$, the percentage estimation error $\hat{e} = \frac{\hat{qf}() - qf()}{qf()}$ is computed, and the provider incurs a fine that is proportional to $\hat{e} - \delta$, whenever $\hat{e} > \delta$.

As a result, the provider's chance of getting away with an incorrect estimate (and hence, a reduced penalty) depends on the authority's budget and ability to monitor effectively.

In this scheme, the consumer relies on the authority for control. In case of dispute of past transactions, the authority is obliged to perform a check on a past quality value, which may require enabling infrastructure. For completeness, this amounts to querying a past state of the D_r database – which is feasible if D_r is a standard transactional DBMS with logging capabilities.

[2]We are implicitly assuming, for the purpose of the example, that the monitoring cost in this case is proportional to the number of items retrieved from D_r.

4. PROVIDER COMPLIANCE MODEL

In this section we analyze the issues associated to enforcing an agreement, from the provider's perspective. The quality-constrained data provisioning problem can be described according to a simple "monitor-assess-repair" reactive model:

monitor: Firstly, the provider must compute the value of quality functions every time it receives a query. Since some of the function parameters may not be available, i.e., $Q(D_r)$, they must be estimated;

assess: Secondly, the provider must estimate the penalty associated with the result set for the query;

repair: Thirdly, it must determine the most cost-effective repair actions to be executed in order to move the state of its data set towards compliance.

With reference to completeness and consistency, the model is instantiated as follows.

Detection. We denote with $\overline{D}_Q$ the quantity we wish to monitor. For completeness, this quantity is

$$\overline{D}_Q, compl = Q(D_r) \setminus Q(D)$$

This set contains all the items that the user would have obtained by issuing Q to D_r, but are instead missing from $Q(D)$. These are therefore the items responsible for the penalties incurred when returning $Q(D)$. For consistency, $\overline{D}_Q$ is the set of items that were expected to be consistent, but are not:

$$\overline{D}_Q, cons = Q(D) \setminus acc(val, Q(D), D_r, c_0)$$

Repair. For completeness, the only repair procedure consists in obtaining new data items from D_r. For consistency, the procedure may perform validation on items whose consistency is unknown, and apply algorithms to enforce consistency (for instance, by correcting data or obtaining new versions from a third party source).

Costing. The third step is the choice of a provider cost model, which includes a *monitoring* component $cost_m(Q, D)$, a *repair* component $cost_r(D)$, and the *value-adding* component $cost_{va}(d)$ of expending local resources in order to prepare any $d \in D$ for delivery.[3] For completeness, the first two correspond to the cost of computing $\overline{D}_Q, compl$ and the cost of obtaining a new item d from D_r, respectively.

Compliance strategies. Next, the provider must identify a strategy for activating repair procedures given the price and penalty information from the agreement, the observed violations, the cost model, and a goal. An obvious general provider goal is to avoid penalties that erode profit, by incurring the minimal repair cost. For completeness, this translates into the strategy of obtaining the smallest set of items from D_r, which restores the required completeness levels. As noted, D may be a small subset of D_r, however if it contains the items that are most likely to be requested in the future, these may be sufficient to satisfy the constraints for most of the user queries. Therefore, a sensible approach is to use the history of past queries to estimate the likelihood of any item in D_r being requested in the future. Estimating future request probability is clearly more effective than a simpler greedy strategy, which would acquire only the items

[3]This could for example be the cost of producing a review for a new article.

that are missing from the current query, some of which will not be requested again.
Regardless on the specific choice of estimator, the precision (i.e., confidence level) associated to the estimate depends on the length of query history: for a new agreement or a new user, the provider will be able to do little more than repairing based on the current query. Various statistical models can be developed to estimate such probability, and it is not the purpose of this work to survey them. Simple estimators include the frequency of past requests, which assume that the past popularity of an item is an indicator of future interest; a mobile average on a limited time window, which attempts to track the changes in interest; or an estimator based on the hypothesis that the occurrence of a request has a known distribution. We note in passing the similarity between the problem of predicting the request of items that are obtained from reference sources, and the problem of defining cache replacement algorithms: for properties like completeness, similar estimators may be applicable.

4.1 Factors that affect compliance strategies

A number of factors complicate the choice and implementation of a strategy:

- **Instant repair:** is it feasible for the provider to obtain new items from D_r, and use them to repair the current result set? When this is not possible, current penalties are inevitable, and the repair strategy may only focus on avoiding future penalties.

- **Completeness of detection:** has the provider complete knowledge of the quality indicators? The provider must balance the precision of the $\overline{D}_Q$ estimate, with the monitoring cost $cost_m()$ of computing it, and the chance that the verification authority will claim irregularities.

- **Number of quality properties** that appear in a single agreement, or in multiple agreements: the potential interaction between constraints on different properties may complicate the strategy. Consider for instance *currency*, the property of a data value of being correct at a given time[4]. Currency and completeness are not independent, because if we assume that all items in D_r are current, then obtaining a new item from D_r in order to restore completeness, has also the effect of improving currency. On the other hand, given a budget for acquiring new items, there may be a contention between different quality constraints that depend on those items for their satisfaction, breaking the isolation of single-property strategies.

- **Options available for detection and repair:** while for completeness the only repair option is to obtain items from a reference source, multiple such sources may be available, possibly at different costs. Also, other properties may present richer options: validating an item for consistency may involve requesting a correction from a reference source, or performing manual inspection on the item.

[4]The data for an address that changed recently may have been correct before the change occurred, but it has since become obsolete, or non-current, until it is updated.

- **Notification of updates to a reference source:** for properties whose repair actions depend on a reference source, it matters whether the provider is notified of any update to the source. For instance, if completeness is estimated by periodically sampling D_r, then the estimate is affected by the frequency of updates to D_r.

- **Shape of cost and price functions:** the various cost and price functions listed earlier may depend on the particular choice of item, or may be defined as a function of the size of the result.

5. PROVISIONING WITH COMPLETENESS

In order to provide a concrete example of provisioning with quality, we now illustrate an algorithm for completeness, based on the detection-repair model. With respect to the complicating factors listed above, the assumptions for the algorithm are as follows: instant repair is possible; only one quality property appears (completeness), and the only repair option is to obtain new items from the reference source; there is no notification of updates to the reference; and finally, the price function depends only on the size of the query result.

We first present a baseline algorithm that assumes that the provider has complete and free knowledge of the quality indicators, and then propose a generalization that does not require this assumption.

5.1 Baseline algorithm

The approach is based on the definition of a utility function for a set $D' \subseteq D_r \setminus D$ of currently missing items, and the formulation of a corresponding optimization problem, that can be solved using heuristics. For completeness, the function takes into account the provider cost model and the penalty functions:

$$\begin{aligned} U(D', Q, D, \mathcal{P}) \;=\; & price_b(Q(D) \cup (D' \cap Q(D_r))) \\ & \times (1 - pen_{compl}(Q(D) \cup (D' \cap Q(D_r)), \mathcal{P})) \\ & - cost_r(D') - cost_{av}(D') \end{aligned}$$

In practice, U describes the effect of purchasing set D':

- the base price is increased due to the new items in D'. Notice that there is no guarantee that the algorithm will only purchase items that are missing from the current result. In fact, the heuristic presented later makes a less greedy selection, hoping to improve *future* compliance. Hence, only the items in $D' \cap Q(D_r)$ contribute to the immediate extra reward.

- The penalty is reduced correspondingly (the same observation applies).

- The cost incurred is due to purchasing and adding value to D'.

The optimization problem is designed to limit the risk of purchasing items that may not be needed in the future: we are seeking the subset of $D_r \setminus D$ that maximizes the ratio of utility-to-size:

$$\max_{D' \in D_r \setminus D} \frac{U(D', Q, D, \mathcal{P})}{|D'|}$$

Normalizing by size avoids the effect of an indefinitely increasing utility, which would result in purchasing the largest possible D'. Note that the problem is defined on the entire set of missing items, rather than only on $\overline{D}_Q$, and that it must be solved when Q is computed.

Since we are not making any assumptions on the shape of function U, or of any of its components, a brute-force algorithm that enumerates all possible D' has exponential complexity in $|D_r \setminus D|$. To reduce the complexity, we apply the strategy mentioned in Section 4, using the history of past user queries to estimate the likelihood of a missing item to be requested in the future.

Algorithm 1. For each Q, the provider maintains a count of the frequency of requests of each item $d_i \in \overline{D}_Q$, and requires an estimate of the likelihood of a future request for each d. Since this involves updating the frequency of the items that are actually requested, complete knowledge of D_r is not required. The algorithm starts from an empty set D', and incrementally adds to it in order of estimated likelihood, recording the value of U at each step. In this way, at step i only the most promising of the $\binom{|D_r \setminus D|}{i}$ potential sets is considered. □

Some comments are in order:

- From the definition of the utility function, we note that its components depend not only on the number of items, but also on their choice. This is because we allow the selection of D' to range on the entire set $D_r \setminus D$, rather than only on $\overline{D}_Q$, hence some of the selected items may not reduce the immediate penalty.
- The order defined on the missing items is partial. For instance, after the first query, the best set contains a random selection (of optimal size) of items from $\overline{D}_Q$, because all such items have the same frequency of occurrence. We assume that the items in $\overline{D}_Q$ are preferred over others of the same rank.
- It is worth considering the effect of a *locality principle* on this heuristic, which states that the history of past queries is indeed a good predictor for future queries. Consider what happens when queries are highly localized and an occasional odd query arrives, requesting items never mentioned before. Since these items have a low frequency, they are ranked low relative to others that have been requested in the past multiple times, but are still missing. In this case, the algorithm does not try to repair the current query (which will therefore result in a penalty), but rather it will purchase additional popular items, increasing the expected reward for future queries.
- As noted earlier, initially the limited history of past requests makes the estimators unreliable, yielding items that may in fact never be used again in the future. This confirms the intuition that this agreement model makes frequent and regular consumers more appealing than occasional ones, and suggests that quality agreements are best suited for long-term consumer-provider relationships.

We conclude by noting the effect of updates and inserts into D_r. In this version of the algorithm, even if the provider is not informed of these events (for instance, through some event notification infrastructure), they do not pose problems.

When an update comes to D_r, then D clearly holds a stale copy, of which it is not aware. However, unless there is an explicit currency constraint, this has no consequences on completeness! – this is in fact a case for handling completeness and currency together. When a new insert occurs in D_r, according to our algorithm it may be revealed only through queries of the form $Q(D_r)$. Items that appear in D_r but are never requested, are simply ignored. Items that start being requested after they have been inserted, are handled in the same way as all others.

5.2 Ranking of missing data using query predicates

Given a query of the form $Q = \sigma_p(R)$, our baseline algorithm relies on the *extension* D_r for computing completeness (detection), and for selecting the most promising items to purchase (repair), assuming $Q(D_r)$ known, by simply enumerating the missing items. The algorithm described in this section addresses the problem of performing detection and repair when $Q(D_r)$ is not available.

The idea is to consider only the *conditions* stated in the user queries, and those used by the provider to obtain new items from D_r. We rely on two observations: firstly, that the most popular data are represented at the *intensional* level by the history of user queries; and secondly, that although $Q(D_r)$ is not immediately available, within the limited scope of a specific user query we may still provide a good estimate for the completeness $compl_Q(D, D_r)$, and also determine the conditions corresponding to the most popular items in D_r, for repair.

The algorithm is based on the definitions of *request profiles* and *completeness maps*. Similar in spirit to ordinary database profiles used by relational query optimizers, a request profile records the level of interest for specific data items, and is computed progressively from a history of user queries. The main difference is that, while in ordinary profiles the data points in the histograms are attribute values, in this case they are *query predicates*.

Whereas a request profile records the demand for data, a completeness map represents the available data set, as described by the set of queries issued by the provider to D_r. Intuitively, knowledge of the user requests to the provider and of the provider requests to its suppliers is sufficient to rank the data that the provider is missing, according to the user preferences.

Let $\mathcal{Q} = Q_1, Q_2, \ldots, Q_m$ and $\mathcal{Q}' = Q'_1, Q'_2, \ldots, Q'_n$ be the history of user and provider queries, respectively. We may restrict our attention to select-queries only, of the form $Q = \sigma_p(R)$, ignoring projections; having defined completeness at the granularity of the entire data item, a distinction based on projected attributes is unnecessary. Furthermore, we make the simplifying assumption that selection predicates are conjunctions of elementary conditions that are either (i) expressions involving relational operators *relop* $(=, \leq, \geq)$ on ordered domains, of the form x *relop* c, or (ii) set membership expressions on enumerated sets, i.e., $x \in \{c_1, \ldots, c_n\}$.

Given a relation $R(A_1, \ldots, A_l)$, these definitions are formalized as follows.

DEFINITION 1. *(Request profile)*

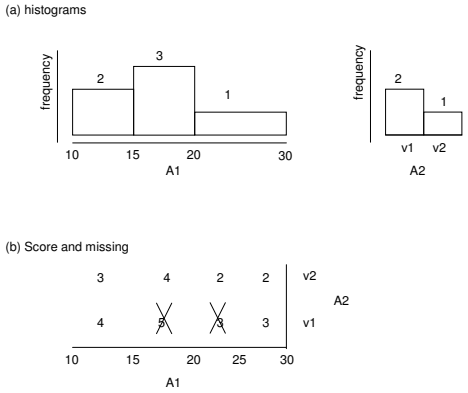


Figure 2: Construction of request profiles

Given set $P_i = \{p_{i1}, \ldots, p_{in_i}\}$ of logically disjoint predicate expressions p_{ij} on A_i, a request profile is a set of l histograms

$$freq_i : P_i \rightarrow \mathcal{N},$$

one for each A_i. Each $freq_i$ maps P_i onto its frequency of occurrence as observed in the query history $\mathcal{Q}$.

For example, suppose that D is described by the single table $R(A_1, A_2)$, where A_1 ranges over the positive integers, and A_2 ranges over the finite set $\{v_1, \ldots, v_n\}$. Let the predicates found in the history of queries be:
$p_1 = (A_1 \in [10, 20])$
$p_2 = (A_1 \in [10, 20] \wedge A_2 = v_1)$
$p_3 = (A_1 \in [15, 30] \wedge A_2 = v_2)$
$p_4 = (A_2 \in \{v_1, v_2\})$
We write $[10, 20]$ as a shorthand for $A_1 \in [10, 20]$. The histograms are constructed as follows:

1. Q_1 carries predicate $[10, 20]$, so $freq_1([10, 20]) = 1$.
2. After Q_2, $freq_1([10, 20]) = 2$ and $freq_2(v_1) = 1$.
3. Q_3 causes the $[10, 20]$ interval to be *refined* into the adjacent disjoint intervals $[10, 15]$, $[15, 20]$ and $[20, 30]$, associating a frequency to each:
   $freq_1([10, 15]) = 2$,
   $freq_1([15, 20]) = 3$,
   $freq_1([20, 30]) = 1$
   (partitioning two overlapping intervals into disjoint intervals can be accomplished easily). Also, now $freq_2(v_1) = 2$ and $freq_2(v_2) = 1$.

The resulting histograms are illustrated in Figure 2 (a). Notice that these histograms are independent of each other: for simplicity, we do not account for the co-occurence of some of the predicates within the same query. Also, by construction the histograms only include the predicates that appear in the queries, rather than all possible combinations for the value set of each attribute.

Definition 2. *(Space of predicates)*
Given the sets $\{P_1, \ldots, P_l\}$ of predicate expressions for a request profile, the space of predicates $\mathcal{P}$ is the set of all vectors of the form $\mathbf{p} = (x_1, \ldots, x_l)$, with $x_k \in P_i$.

In the example, $\mathcal{P}$ includes $([10, 15], v_1)$, $([10, 15], v_2)$, $([15, 20], v_1)$ and so forth. We describe the popularity of a combination $\mathbf{p} \in \mathcal{P}$ of predicates using a syntectic score value:

$$score(\mathbf{p}) = \sum_{i:1..l} freq_i(p[i])$$

i.e., $score([15, 20], v_1) = 5$, $score([20, 30], v_2) = 2$, etc.

This score, however, is oblivious of the data from D_r that has already been purchased. Thus, it is complemented by the partial information on the completeness D relative to D_r:

Definition 3. *(Completeness map)*
Given $\mathbf{p} = (x_1, \ldots, x_l) \in \mathcal{P}$, let $p = x_1 \wedge \cdots \wedge x_l$ be a predicate. A completeness map is described by boolean function

$$missing(p) \in \{true, false\}$$

defined on the space of predicates, such that

$$missing(p) = true \text{ iff } Q_p(D_r) \subseteq D.$$

Assuming $missing(p) = true$ for all p initially, the map is updated using the history $\mathcal{Q}'$ of data purchases. For instance, let $p'_1 = (A_1 \in [15, 25] \wedge A_2 = v_1)$ be the predicate for Q'_1. As shown in Figure 2 (b), first p'_1 is used to further refine the histogram for A_1, adding the interval boundary 25. The corresponding *score* table is updated as a consequence. Splitting $[15, 25]$ into $[15, 20]$ and $[20, 25]$ aligns this interval with the existing histograms. Then, we set $missing([15, 20], v_1) = missing([20, 25], v_1) = 0$. In practice, we mark selected points in the space of predicates, which represent conjunctions that have already been used to purchase new data.

The rank of a point $\mathbf{p}$ representing missing data is simply the product

$$rank(\mathbf{p}) = score(\mathbf{p}) \ . \ missing(\mathbf{p})$$

This ranking provides a preference only among the predicates that describe popular items that are still missing (the others have rank 0), and replaces the simpler repair criterion used in the baseline version of our algorithm. In the example, the combination $([10, 15], v_1)$ is the most popular, since $([15, 20], v_1)$ is not missing.
It is worth mentioning that the operation of histogram refinement that may follow each user query, also requires reevaluating the *missing* function. This is simple, however, since each new sub-interval inherits the *missing* values found in the parent interval (this is left to intuition).

Algorithm INTENSIONAL DATA RANKING
Given relation $R(A_1, \ldots, A_l)$:

For user query $Q = \sigma_p(R)$:
begin
  for each A_i **do** {
    Let p_i be the conjunction of terms from p on A_i;
    affectedPoints = $refine(P_i, p_i)$;
    $updateHistogram(freq_i(p_i))$;
  }
  for each **p** $\in$ affectedPoints **do** $updateScore(\mathbf{p})$;
  for each **p** $\in \mathcal{P}$ **do** $rank(\mathbf{p}) = score(\mathbf{p}) \times missing(\mathbf{p})$;
end

For provider query $Q' = \sigma_{p'}(R)$ issued to D_r:
begin
  for each A_i **do** {
    Let p'_i be the conjunction of terms from p' on A_i;
    affectedPoints = $refine(P_i, p'_i)$;
    for each **p** $\in$ affectedPoints **do** $updateScore(\mathbf{p})$;
    newPoints = $computeNewPoints(p'_i)$;
  for each **p** $\in$ newPoints **do** $missing(\mathbf{p}) = false$;
end

Figure 3: Ranking algorithm

The ranking algorithm is summarized in Figure 3. Function $refine()$ increases the number of intervals in the histogram, and returns the points in the space of predicates that are affected by this operation. For these points, the score is updated prior to computing their rank. Upon issuing Q' to D_r, the provider must additionally compute the new points in the space of predicates corresponding to the query predicate, as shown earlier in the example, and reset their missing flag.

Tables *score* and *missing* can also be used as a basis to provide various estimates of completeness for a query $Q = \sigma_p$:

$$compl_{\sigma_p}(D, D_r) = \frac{|\sigma_p(D) \cap \sigma_p(D_r)|}{|\sigma_p(D_r)|}$$

For example, given predicate $p = [10, 20]$ with the situation illustrated in Figure 2, one may view intervals as discrete elements, regardless of their width, and observe that p corresponds to the "slice" of the *score* table that includes 4 predicate combinations. Since only one of these is not missing, one may estimate $compl_p(D, D_r) = 0.25$. Alternatively, the width of each of the intervals involved or other weight factors can be taken into account, yielding different estimates.

6. REFERENCE ARCHITECTURE

The architecture sketched in Figure 4 consists of a quality management module that contains the key components described in the previous sections. When a user query comes through the client service interface, it is processed as usual; before the result is returned, it is intercepted and passed to the quality management module, and query post-processing occurs. Using the interceptor pattern ensures that no changes are required to the query processor.

With reference to completeness, query post-processing proceeds as follows. The detection component is in charge of monitoring the completeness indicator and of computing the quality value, estimating the penalty associated with the query result. As explained in the previous section, this is done by querying the profile manager, which controls the completeness maps. The user query is also used by the profile manager to update the request profiles.

The strategy manager then uses this information to compute the utility function and to setup the optimization problem. Again, the profile manager is a critical component, in that it provides the ranked predicates that correspond to the most interesting new items. At this point, the repair actions consist of one or more queries to D_r, issued by the repair component using the pull interface of the gateway.

If the "instant repair" option is available, described in Section 4.1, then the new data items are prepared for immediate delivery and added to the original result set. Additionally, the queries are passed to the profile manager, which updates the completeness map.

If a push interface is active, then any update to D_r is propagated not only to D, but also to the profile manager, which may use it to update the completeness estimates. At the end of post-processing the final price is computed, taking self-assessed penalties into account, and the result is returned to the user. In the figure, an agreement interface is also shown as part of the quality management module, to indicate the channel used for agreement negotiation, which results in the configuration of the module components.

7. FURTHER WORK AND CONCLUSIONS

We have presented a simple model for the definition of formal agreements between data providers and consumers regarding the quality levels of data, illustrating the provider's problem space and showing an algorithm for dealing with the completeness property as a special case. Finally, we have described a reference provider architecture for enforcing quality agreements, that is respectful of the existing query processing architecture.

This work is at its initial stages and can be extended in many directions, adding elements that may affect our initial assessment of the agreement model and of the providers' strategies. Firstly, we believe that an experimental evaluation of the presented approaches for completeness, and a prototype implementation of the architecture, may provide an insight into their practicality. Secondly, the overall framework and architecture must be tested by considering additional properties: are there suitable architectural patterns for dealing with constraints on multiple quality properties, and how would the provider define strategies that involve interplay between properties, i.e., between completeness and currency? In the same vein, dealing with multiple agreements with overlapping constraints may intuitively make the provider's strategy more cost-effective, in ways that need to be investigated.

8. REFERENCES

[1] A.Borthwick, M.Buechi, and A.Goldberg. Key concepts in the choicemaker 2 record matching system. In *Procs. First Workshop on Data Cleaning, Record Linkage, and Object Consolidation, in conjunction with KDD 2003*, Washington, DC, July 2003.

[2] A.Dan, D.Davis, R.Kearney, A.Keller, R.King, D.Klueber, H.Ludwig, M.Polan, M.Spreitzer, and

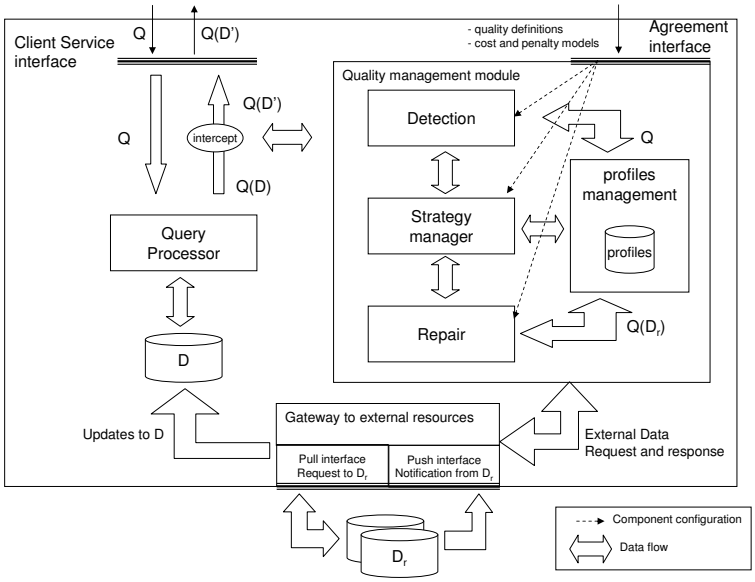


Figure 4: Reference architecture

A.Youssef. Web services on demand: WSLA-driven automated management. *IBM Systems Journal*, 43(1), 2004.

[3] A.Motro, P.Anokhin, and A.C. Acar. Utility-based resolution of data inconsistencies. In Felix Naumann and Monica Scannapieco, editors, *International Workshop on Information Quality in Information Systems 2004 (IQIS'04)*, Paris, France, June 2004. ACM.

[4] C. Cappiello, C. Francalanci, P. Missier, B. Pernici, P. Plebani, M. Scannapieco, and A. Virgillito. Presentation of metadata and of the quality certificate. Deliverable dl2, The DaQuinCis project, 2003.

[5] D.Ballou and G.K.Tayi. Methodology for allocating resources for data quality enhancement. In *Communications of the ACM*, volume 32. ACM, March 1989.

[6] D.Ballou and H.Pazer. Designing information systems to optimize the accuracy-timeliness tradeoff. *Information Systems research*, 6(1), 1995.

[7] D.Ballou, R.Wang, H.Pazer, and G.K.Tayi. Modelling information manufacturing systems to determine information product quality. *Journal of Management Sciences*, 44(4), April 1998.

[8] M.G. Elfeky, A.K. Elmagarmid, and V.S. Verykios. Tailor: a record linkage tool box. In *Proceedings of the 18th International Conference on Data Engineering (ICDE 2002)*, San Jose, CA, Feb. 2002. IEEE Computer Society.

[9] L. P. English. *Improving data warehouse and business information quality: methods for reducing costs and increasing profits*. John Wiley & Sons, 1 edition, March 1999. ISBN: 0471253839.

[10] I.P. Fellegi and A.B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64, 1969.

[11] F.Naumann, J.C.Freytag, and U.Leser. Completeness of integrated information sources. *Information Systems*, 29(7):583–615, 2004.

[12] S. Kalepu, S. Krishnaswarmy, and S.W. Loke. Verity: a qos metric for selecting web services and providers. In *Proceedings of 4th International Conference on Web Information Systems Engineering Workshops (WISEW'03)*. IEEE Computer Society Press, 2004.

[13] P. Missier and C. Batini. A multidimensional model for information quality in cooperative information systems. In M. Helfert M. Eppler, editor, *Proceedings of the Eight International Conference on Information Quality (ICIQ-03)*, 2003.

[14] M.Lenzerini. Data integration: A theoretical perspective. In *Principles Of Database Systems*, pages 233–246, 2002.

[15] M.Scannapieco, A.Virgillito, C.Marchetti, M.Mecella, and R.Baldoni. The architecture: a platform for exchanging and improving data quality in cooperative information systems. *Inf. Syst.*, 29(7):551–582, 2004.

[16] M.Scannapieco and C.Batini. Completeness in the relational model: a comprehensive framework. In *Procs. 9th International Conference on Information Quality, ICIQ 2004, Cambridge, Ma*, 2004.

[17] F. Naumann, U.Leser, and J.C.Freytag. Quality-driven integration of heterogenous information systems. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases*, pages 447–458, Edinburgh, Scotland, UK, September 1999. Morgan Kaufmann.

[18] T.C. Redman. *Data quality for the information age*. Artech House, 1996.

[19] R.Y.Wang, M.Ziad, and Y.W.Lee. *Data quality*. Advances in Database Systems. Kluwer Academic Publishers, 2001.

[20] J. Skene, D. D.Lamanna, and W. Emmerich. Precise service level agreements. In *Proceedings of 26th International Conference on Software Engineering (ICSE'04)*. IEEE Computer Society Press, 2004.

[21] Y. Wand and R.Wang. Anchoring data quality

dimensions in ontological foundations. *Communications of the ACM*, 39(11), 1996.

[22] R. Wang. A product perspective on total data quality management. *Communications of the ACM*, 41(2), February 1998.

[23] R. Y. Wang, M. Ziad, and G. Shankaranarayanan. IP-MAP: representing the manufacture of an information product. In *Proceedings of the Eight International Conference on Information Quality (ICIQ-00)*, Cambridge, MA., November 2000.

[24] R.Y. Wang and D.M. Strong. Beyond accuracy: what data quality means to data consumers. *Journal of Management Information Systems*, 12(4), 1996.

Making Quality Count in Biological Data Sources

Alexandra Martinez
Dept of Computer & Information Science & Engineering
University of Florida
Gainesville, FL 32611 USA
+1 (352) 392-1200
amartine@cise.ufl.edu

Joachim Hammer
Dept of Computer & Information Science & Engineering
University of Florida
Gainesville, FL 32611 USA
+1 (352) 392-1200
jhammer@cise.ufl.edu

ABSTRACT

We propose an extension to the semistructured data model that captures and integrates information about the quality of the stored data. Specifically, we describe the main challenges involved in measuring and representing data quality, and how we addressed them. These challenges include extending an existing data model to include quality metadata, identifying useful quality measures, and devising a way to compute and update the value of the quality measures as data is queried and updated. Although our approach can be generalized to various other domains, it is currently aimed at describing the quality of biological data sources. We illustrate the benefits of our model using several examples from biological databases.

1. INTRODUCTION

The rapid and continuous increase in the amount of biological information available (both experimental and derived) has brought concern over the perceived quality of existing data. Analysis and processing of faulty data produces misleading results and could hamper scientific progress. At the same time, most research efforts in the area of data quality assessment have been geared towards data quality problems in enterprise data warehousing, and less work has been aimed at analyzing and recording the data quality of biological databases. Hence there is a need for quality assessment measures in biological data sources and we propose a model for assessing and recording quality information in biological data sources.

We define data quality as a measure of the trustworthiness of the data. Data that is trustworthy conforms to reality, and therefore can be relied upon for any decision-making, analysis, or to derive new knowledge. Our definition of data quality is consistent with those that regard data quality as the degree of agreement between data views presented by an information system and that same data in the real-world [21]. However, the trustworthiness of the data is still a very abstract and broad concept, which is difficult to measure. In our approach we decided to decompose the notion of trustworthiness into different metrics each of which is a quantifiable value. In this sense, we also agree with the notion of data quality as a multi-dimensional concept [22, 26].

There has also been a growing interest from the database community in a special type of data known as *semistructured data* (or *unstructured data*), which is commonly defined as "schemaless" or "self-describing" data [1,5]. Semistructured data differs from the kind of data handled by relational databases in that it does not have a rigid structure; hence the interest in finding new ways of modeling and managing this type of data. The use of semistructured-like models in the biological context started off with the ACeDB system (which still requires a schema, but imposes only weak constraints) [1,5], and has recently gained popularity with the development of XML-based languages for biology such as BioML [25], BSML [4], AGAVE [2], and XEMBL [28], just to mention some. Currently, most biological repositories (including GenBank [10], EMBL [9], and DDBJ [8]) can export their data in XML format, using some of the existing XML languages. This trend suggests that semistructured data models (XML being one of them) are expressive and flexible enough to adequately represent biological data, which characterizes by having large variability and missing data.

The main contributions of our work are: 1) The definition of six quality measures that are meaningful in the biological domain; 2) the conceptual integration of these measures into a data model suitable for representing both biological data and quality measures; and 3) the description of how these quality measures change as data is queried or updated (according to the operations of our data model), and how they also affect the query result.

The rest of the paper is organized as follows. Section 2 presents some related work. Section 3 presents our approach, particularly how we measure the quality of biological data, and how we integrate the quality information into a data model fitting both data and quality metadata. It also describes how the operations in the data model will change to encompass our quality metadata. Section 4 illustrates the use of our quality measures with several examples taken from RefSeq [20], a popular genomic data source. In Section 5 we outline our conclusions and suggestions for future work.

2. RELATED WORK

Research related to our work can roughly be divided into three areas: *Information Quality* research aimed at defining, measuring and evaluating the quality of data, *Data Quality* research in cooperative information systems, and research on *semistructured data models*. We briefly summarize the state-of-the-art in each area below.

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

Information Quality (IQ) is commonly defined as "fitness for use" (i.e., achieving a level of data quality that is sufficient from the perspective of its users) [3]. Numerous models, evaluation methodologies, and improvement techniques have been developed in the area of Information Quality [12,13,16,24,27]. Wang et al. [27] proposed an attribute-based model to tag data with quality indicators, which characterize the data and its manufacturing process. They suggest a hierarchy of data quality dimensions with four major dimensions: accessibility, interpretability, usefulness, and believability. Each of these dimensions is split into other factors such as availability, relevancy, accuracy, credibility, consistency, completeness, timeliness, and volatility. Mihaila et al. [16] identified four Quality of Data parameters: completeness, recency, frequency of updates, and granularity. Lee et al. [12] distinguished five dimensions of data quality: accessibility, relevancy, timeliness, completeness, and accuracy; each considered a performance goal of the data production process. Lee et al. [13] developed a methodology for IQ assessments and benchmarks called AIM quality. This methodology is based on a set of intrinsic, contextual, representational, accessibility IQ dimensions which are important to information consumers. These dimensions were first devised by Strong et al. [24] as categories for high-quality data. All these research efforts offer valuable contributions for better understanding data quality problems and challenges, but they often lack the pragmatic component which will ultimately allow users to make decisions about the quality of the data based on quantitative measures.

More recently, Data Quality has been studied in the context of cooperative information systems [15,23,19,17], where more practical approaches have emerged. Mecella et al. [15] describe a service-based framework for managing data quality in cooperative information systems, based on an XML model for representing and exchanging data and data quality. Scannapieco et al. [23] developed the DaQuinCIS architecture and the D^2Q (Data and Data Quality) model for managing data quality in cooperative information systems. They defined four data quality dimensions: accuracy, completeness, currency, and consistency. Naumann et al. [19] presented a model for determining the completeness (i.e., a combination of density and coverage) of a source or combination of sources. Missier et al. [17] defined the notions of quality offer and quality demand within cooperative information systems, and modeled quality profiles as multidimensional date cubes. In spirit, these works are closely related to ours because they seek concrete and systematic means of computing the quality of data. However, our data quality model is not aimed for cooperative information systems but rather for single biological information systems. In the biological context, data quality has been addressed by Müller et al. [18], who examined the production process of genome data and identified common types of data errors.

In the area of semistructured data, several models have been proposed [7,5,1]. Most of them, however, use the same underlying representation: a graph-like or tree-like structure [5]. For example, Abiteboul et al. [1] use an edge-labeled graph to represent semistructured data. UnQL and LORE are based on an edge-labeled tree representation [6,14]. Calvanese et al. [7] use the basic data model for semi-structured data (called BDFS) in which both databases and schemas are represented as graphs. The work by Scannapieco et al. [23] is also relevant here because they provide a strong association between their D^2Q schemas and XML schemas, thus proving XML convenient for representing both data and data quality.

3. APPROACH

Our work addresses the lack of a formal model for measuring and representing quality information in biological data sources. Several challenges must be overcome. The first one is to identify a core set of quality measures which provides a framework for assessing the quality of biological data. A second challenge is to devise a data model for recording and maintaining biological data that facilitates the integration of quality measures into the model. A third challenge is to determine the effect of the operations defined in the data model on the quality measures, and the influence of the quality measures on the result of the operations. We address our approach in response to each of these challenges in the following subsections.

3.1 Measuring the Quality of Data

In order to identify useful quality measures, we define criteria that the measures should satisfy: *biologically-relevant*, *objective*, and *easy-to-compute*. Since our work is framed within a biological context, our measures should be meaningful to biologists so they can use these measures to discern among data of different quality levels. Additionally, our measures should be objective, meaning that there is no room for ambiguous interpretation (i.e., subjective appraisal) when assessing the value of a measure. Last but not least, our measures should be easy-to-compute in a real system that efficiently handles quality-augmented data in a scenario where data is constantly being updated.

Using the above constraints, we have identified the following six quality measures: Stability, Density, Time since Last Update, Redundancy, Correctness, and Usefulness. We have classified them into two sets: primary measures, and derived measures. Primary measures are independent of each other, while derived measures depend on one or more primary measures. We provide an intuitive description for each of them below. Formal definitions will be given shortly.

3.1.1 Primary Measures

- *Stability* indicates whether the data is undergoing a period of change. This measure is obtained by computing a weighted average of the magnitude of changes applied to a data item since its creation, where more weight is assigned to recent changes. Changes that happened long time ago will have a small weight and thus will not affect the current stability of the data item significantly. On the other hand, a recent change to the data item will have a large impact over its current stability value. A value of 1 corresponds to maximum stability while 0 corresponds to maximum instability.

- *Density* indicates the number of components (or attributes) that describe a data item. For instance, a data item that consists of a single value is less dense than a data item involving several values. Note that density does not take into account the number of bytes needed to store a data item, since each data item is considered a data unit regardless of the space it takes. The lowest density value a data item can have is 1, but there is no upper limit. Our density measure is comparable to the combination of density and coverage

described in [19] since we account for both the intension and extension of a data source. However, our density measure is formulated in a different way because we only consider the density of data items within a single source, as opposed to [19]'s mediator system which involves many sources and uses them to build a normalizing factor based on a universal relation.

- *Time since Last Update* (*TsLU*) measures the time elapsed since a data item was last updated. Note that TsLU is not the same as the age[*] of the data item. A TsLU value of 0 denotes recently updated data, whereas higher values denote elder data. Our TsLU measure differs from the currency quality dimension defined in [23] in that we do not assume knowledge about the time when a data value changes in the real world. Actually, in our biological context, real world data (such as DNA sequences) are supposed to remain unchanged; so all changes are due to a human curation process that aims to correct errors introduced during data *collection*.
- *Redundancy* measures the fraction of redundant information contained in a data item. A data item contains redundant information if any of its components (sub-items) is redundant. A data item is redundant if it represents the same real world object as any other data item in the data source. A redundancy value of 0 indicates the data item contains no redundant information, whereas a value close to 1 indicates that a large fraction of the information in the data item is redundant. Redundancy is related to the consistency quality dimension described in [23] in that redundant data items usually are also inconsistent (i.e., their data values conflict with each other). However, [23] only defines consistency for attribute values within the same schema element (entity) instance, but our redundancy measure is defined also across instances. In the biological context, redundancy can arise due to multiple submissions of the same "real world" data (e.g. a DNA or protein sequence) by different authors, which usually differ in their annotations.

3.1.2 Derived Measures

- *Correctness* is a measure of the accuracy of a data item. It can also be regarded as the degree of confidence that the data item represents true information. A correctness value of 1 means the data item is completely accurate (i.e., we are 100% confident that the data is true), while a correctness value of 0 means the data is wrong (i.e., we cannot have any confidence about the accuracy of the data). At an abstract level, our correctness measure expresses the same notion as the accuracy dimension described in [23], but it differs in the way it is computed. In [23], it is proposed to use a distance function between the value stored at the data source and the value considered as 'correct'. In biology, however, we cannot make the assumption that such 'correct' value is available, due to the uncertainty associated to the data collection process and the lack of understanding about many biological processes. Therefore, a different approach is needed to estimate the correctness value of biological data. The approach we propose is to use a combination of the stability and age of the data item.
- *Usefulness* measures the utility of a data item as a function of the density, correctness, and redundancy of its components. Particularly, the usefulness of a data item is the ratio between the amount of *information* provided by the data item and its total amount of data. The amount of *information* conveyed by a data item is the amount of non-redundant correct data contained in the data item. The total amount of data contained in a data item is the sum over the density of its components. A usefulness value of 1 means the data item is completely useful (i.e., all of its data is non-redundant and correct), while a value of 0 means the data item is useless.

We believe that the proposed derived measures provide the means for grasping important quality aspects of the data, while the simple primary measures serve to construct more complex quality features. Other measures can be obtained from different combinations of the primary and/or derived quality measures defined here. Therefore, our set of quality measures may be enlarged and improved if other measures are considered to be more meaningful in certain domains.

3.2 Integrating Quality Metadata and Data

By now we have a framework for measuring the quality of data along various dimensions. Our next challenge is the conceptual integration of these quality measures into a data model for validation purposes.

3.2.1 Choosing the right Data Model

We start by selecting a model to represent semistructured data, and we choose the graph model proposed by Abiteboul et al. [1]. The main reasons for choosing this data model are its flexibility (it can represent data lacking a fixed schema), its simplicity (syntactic constructions are simple and easy to understand), and its fitness for representing biological data (can handle data with missing values and high variability).

Roughly speaking, the model by Abiteboul et al. [1] is an edge-labeled directed graph that represents semistructured data expressions (called *ssd-expressions*). The following syntax taken from [1] describes how *ssd-expressions* can be constructed:

<*ssd-expr*> ::= <*value*> | oid <*value*> | oid
<*value*> ::= atomicvalue | <*complexvalue*>
<*complexvalue*> ::= {label : <*ssd-expr*>, …, label : <*ssd-expr*>}

Here, atomic values are either numbers or strings, and labels are strings of ASCII characters. Object identifiers (oid) are labels bound to the <*value*> that follows them, and we denote them with an ampersand prefix (e.g. &10 or &s1). We illustrate the use of this syntax in Figure 1, where a fragment of the RefSeq record NM_128079 is represented as an ssd-expression. Figure 2 shows the equivalent graph representation of the ssd-expression in Figure 1. The complete NM_128079 record in the original RefSeq format is shown in Figure 3. Next, we extend this data model to include our quality metadata.

[*] Age is the time elapsed since the data item was created.

```
{
  Locus : {
            accession : "NM_128079",
            length-bp : 1356,
            biomol : "mRNA",
            div : "PLN",
            update-date : "25-JAN-2005"
          },
  Definition : "Arabidopsis thaliana protein",
  Version : {
              id : "NM_128079.4",
              gi : 42569303
            },
  Source : {
            Taxname : "Arabidopsis thaliana",
            Organism : {
                         Orgname : {
                                     genus : "Arabidopsis",
                                     species : "thaliana"
                                   },
                         Lineage : "Eukaryota; Viridiplantae; Streptophyta…"
                        }
           }
}
```

Figure 1. Fragment of a RefSeq record represented with the syntax for *ssd-expressions*.

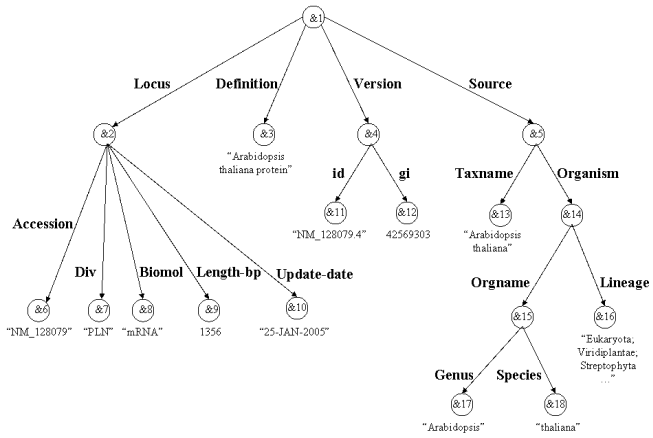


Figure 2. Edge-labeled graph representation of the RefSeq record from Figure 1.

3.2.2 Augmenting the Data Model with Quality Metadata

In order to incorporate our quality measures into the data model, we regard the set of all quality measures of a data item as its *quality metadata*. At a logical level, we represent *quality metadata* as a vector that has one entry (or dimension) per quality measure. Values stored in these entries are the scores associated to each of the quality measures; i.e.,

$$V_Q \equiv [Q_S, Q_D, Q_T, Q_R, Q_C, Q_U]$$

where Q_S, Q_C, Q_D, Q_T, Q_R, and Q_U are atomic objects that give a score for the quality measures *Stability*, *Density*, *TsLU*, *Redundancy*, *Correctness*, and *Usefulness,* respectively.

Using the syntax of our data model (*ssd-expressions*), *quality metadata* can be expressed as a special complex value having one label per quality measure and atomic values (measures scores) following the labels; i.e.,

$$\{S : Q_S, D : Q_D, T : Q_T, R : Q_R, C : Q_C, U : Q_U\}$$

where *S*, *D*, *T*, *R*, *C*, and *U* are labels that represent the *Stability*, *Density*, *TsLU*, *Redundancy*, *Correctness*, and *Usefulness* quality measures, respectively. Q_S, Q_D, Q_T, Q_R, Q_C, and Q_U are atomic values that confer a score to each quality measures. This way of representing the quality metadata is consistent with the data model.

The next issue we need to address is where to place the quality metadata associated to a data item. A data item denotes any of the following syntactic constructions: <*complexvalue*>, <*atomicvalue*>, or <*value*>; which would correspond to a node in the graph representation of the data model. Hence we could place a data item's quality metadata either at the data node or at the edge (i.e., label) coming into the data node. In what follows, we assume that the quality metadata is part of the data item's incoming edge. Making the quality metadata part of the edge (or label) entails some modifications to the original syntax from Section 3.2.1. The following extended syntax incorporates quality metadata into the data model for *ssd-expressions*:

<*ssd-expr*> ::= <*label*> : <*value*> | <*label*> : oid <*value*> |
<*label*> : oid

<*value*> ::= <*atomicvalue*> | <*complexvalue*>

<*atomicvalue*> ::= number | string

<*complexvalue*> ::= {<*ssd-expr*>, …, <*ssd-expr*>}

<*label*> ::= (labelstring, <*qmd*>)

<*qmd*> ::= {labelstring : number, …, labelstring : number}

Note that we have added a general construction for quality metadata (<*qmd*>), which resembles a complex value structure but has simplified labels and values. We have also added a rule for atomic values that allows us to distinguish between numbers and strings; which is needed in the <*qmd*> definition. We augmented the label's syntax so that it can hold quality metadata, and changed the placement of labels in the grammar so that every ssd-expression is forced to have a label rather than only the ssd-expressions within a complex value. Figure 4 shows how the RefSeq record from Figure 3 is represented using our quality-augmented syntax. Labels are merely illustrative: their names were created from the original record's field tags, and their quality metadata is symbolized with a letter 'q' (the reader should interpret this as the actual <*qmd*> structure $\{S : Q_S, D : Q_D, \ldots, U : Q_U\}$).

The semantics of our quality-augmented labels is as follows. If *l* is a label (or edge) with source node *s*(*l*) and target node *t*(*l*), then the quality metadata of *l*, denoted by *q*(*l*), pertain to the data represented by node *t*(*l*) and all its descendant nodes. If *t*(*l*) is a leaf node (i.e., atomic value), then *q*(*l*) refers to the quality metadata of the atomic value stored at that leaf. If *t*(*l*) is an internal node (i.e., complex value), then *q*(*l*) comprises the quality metadata of the complex value represented by the subtree rooted at that node.

```
LOCUS       NM_128079              1356 bp    mRNA    linear   PLN 25-JAN-2005
DEFINITION  Arabidopsis thaliana protein kinase family protein (At2g25220)
            mRNA, complete cds.
ACCESSION   NM_128079
VERSION     NM_128079.4  GI:42569303
KEYWORDS    .
SOURCE      Arabidopsis thaliana (thale cress)
  ORGANISM  Arabidopsis thaliana
            Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
            Spermatophyta; Magnoliophyta; eudicotyledons; core eudicots;
            rosids; eurosids II; Brassicales; Brassicaceae; Arabidopsis.
COMMENT     PROVISIONAL REFSEQ: This record has not yet been subject to final
            NCBI review. This record is derived from an annotated genomic
            sequence (NC_003071). The reference sequence was derived from
            mrna.At2g25220.1.
            On Feb 17, 2004 this sequence version replaced gi:30682693.
FEATURES             Location/Qualifiers
     source          1..1356
                     /organism="Arabidopsis thaliana"
                     /mol_type="mRNA"
                     /db_xref="taxon:3702"
                     /chromosome="2"
                     /map="unknown"
                     /clone="CHR2v01212004"
                     /ecotype="Columbia"
     gene            1..1356
                     /locus_tag="At2g25220"
                     /note="synonym: T22F11.19; protein kinase family protein"
                     /db_xref="GeneID:817060"
     CDS             1..1152
                     /locus_tag="At2g25220"
                     /note="contains protein kinase domain, Pfam:PF00069;
                     go_function: kinase activity [goid 0016301]"
                     /codon_start=1
                     /product="protein kinase family protein"
                     /protein_id="NP_180094.2"
                     /db_xref="GI:42569304"
                     /db_xref="GeneID:817060"
                     /translation="MGSGEEDRFDAHKKLLIGLIISFSSLGLIILFCFGFWVYRKNQS
                     PKSINNSDSESGNSFSLLMRRLGSIKTQRRTSIQKGYVQFFDIKTLEKATGGFKESSV
                     IGQGGFGCVYKGCLDNNVKAAVKKIENVSQEAKREFQNEVDLLSKIHHSNVISLLGSA
                     SEINSSFIVYELMEKGSLDEQLHGPSRGSALTWHMRMKIALDTARGLEYLHEHCRPPV
                     IHRDLKSSNILLDSSFNAKISDFGLAVSLDEHGKNNIKLSGTLGYVAPEYLLDGKLTD
                     KSDVYAFGVVLLELLLGRRPVEKLTPAQCQSLVTWAMPQLTDRSKLPNIVDAVIKDTM
                     DLKHLYQVAAMAVLCVQPEPSYRPLITDVLHSLVPLVPVELGGTLRLTR"
ORIGIN
        1 atgggaagtg gtgaagaaga tagatttgat gctcataaga aacttctgat tggtctcata
       61 atcagtttct cttctcttgg ccttataatc ttgttctgtt ttggcttttg ggtttatcgc
      121 aagaaccaat ctccaaaatc catcaacaac tcagattctg agagtgggaa ttcattttcc
      181 ttgttaatga gacgacttgg ctcgattaaa actcagagaa gaacttctat ccaaaagggt
      241 tacgtgcaat ttttcgatat caagaccctc gagaaagcga caggcggttt taaagaaagt
      301 agtgtaatcg gacaaggcgg tttcggatgc gtttacaagg gttgtttgga caataacgtt
      361 aaagcagcgg tcaagaagat cgagaacgtt agccaagaag caaaacgaga atttcagaat
      421 gaagttgact tgttgagcaa gatccatcac tcgaacgtta tatcattgtt gggctctgca
      481 agcgaaatca actcgagttt catcgtttat gagcttatgg agaaaggatc attagatgaa
      541 cagttacatg ggccttctcg tggatcagct ctaacatggc acatgcgtat gaagattgct
      601 cttgatacag ctagaggact agagtatctc catgagcatt gtcgtccacc agttatccac
      661 agagatttga aatcttcgaa tattcttctt gattcttcct tcaacgccaa gatttcagat
      721 ttcggtcttg ctgtatcgct ggatgaacat ggcaagaaca acattaaact ctctgggaca
      781 cttggttatg ttgccccgga atacctcctt gacggaaaac tgacggataa gagtgatgtt
      841 tatgcatttg gggtagttct gcttgaactc ttgttgggta gacgaccagt tgaaaaatta
      901 actccagctc aatgccaatc tcttgtaact tgggcaatgc cacaacttac cgatagatcc
      961 aagcttccaa acattgtgga tgccgttata aaagatacaa tggatctcaa acacttatac
     1021 caggtagcag ccatggctgt gttgtgcgtg cagccagaac caagttaccg gccgttgata
     1081 accgatgttc ttcactcact tgttccactg gttccggtag agctaggagg gactctccgg
     1141 ttaacaagat gattcacaga aacacgccaa aagaaatcca aagccattta gatgattttc
     1201 ttttatcctt tgcctttata ttttttttgta tagggttatg atccactcat ctgaaagttt
     1261 gggggtaaga atgtgagaat ataagttttc agggttgttg agttctatat aattatattt
     1321 gtttcttttt attgtcaaat ataattatat ttttgt
//
```

Figure 3. Record with Accession number NM_128079.4, taken from RefSeq.

```
(Seq-entry,q):{
        (LOCUS,q):{
                (accession,q):"NM_128079",
                (length-bp,q):1356,
                (biomol,q):"mRNA",
                (div,q):"PLN",
                (update-date,q):"25-JAN-2005"
        }
        (DEFINITION,q):"Arabidopsis thaliana protein kinase family protein (At2g25220) mRNA, complete cds."
        (ACCESSION,q):"NM_128079"
        (VERSION,q):{
                (id,q):"NM_128079.4",
                (gi,q):42569303
        }
        (KEYWORDS,q):""    .
        (SOURCE,q):{
                (taxname,q):"Arabidopsis thaliana (thale cress)",
                (organism,q):{
                        (orgname,q):{
                                (genus,q):"Arabidopsis",
                                (species,q):"thaliana"
                        }
                        (lineage,q):"Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
                                Spermatophyta; Magnoliophyta; eudicotyledons; core eudicots;
                                rosids; eurosids II; Brassicales; Brassicaceae; Arabidopsis."
                }
        }
        (COMMENT,q):"PROVISIONAL REFSEQ: This record has not yet been subject to final
            NCBI review. This record is derived from an annotated genomic
            sequence (NC_003071). The reference sequence was derived from
            mrna.At2g25220.1.
            On Feb 17, 2004 this sequence version replaced gi:30682693."
        (FEATURES,q):{
                (source,q):{
                        (location,q):{
                                (from,q):1,
                                (to,q):1356
                        }
                        (qualifiers,q):{
                                (organism,q):"Arabidopsis thaliana",
                                (mol_type,q):"mRNA",
                                (db_xref,q):"taxon:3702",
                                (chromosome,q):"2",
                                (map,q):"unknown",
                                (clone,q):"CHR2v01212004",
                                (ecotype,q):"Columbia"
                        }
                }
                (gene,q):{
                        (location,q):{
                                (from,q):1,
                                (to,q):1356
                        }
                        (qualifiers,q):{
                                (locus_tag,q):"At2g25220",
                                (note,q):"synonym: T22F11.19; protein kinase family protein",
                                (db_xref,q):"GeneID:817060"
                        }
                }
                (CDS,q):{
                        (location,q):{
                                (from,q):1,
                                (to,q):1152
                        }
                        (qualifiers,q):{
                                (locus_tag,q):"At2g25220",
                                (note,q):"contains protein kinase domain, Pfam:PF00069; go_function: kinase activity [goid 0016301]",
                                (codon_start,q):1
                                (product,q):"protein kinase family protein",
                                (protein_id,q):"NP_180094.2",
                                (db_xref,q):"GI:42569304",
                                (db_xref,q):"GeneID:817060",
                                (translation,q):"MGSGEEDRFDAHKKLLIGLIISFSSLGLIILFCFGFWVYRKNQS
                                        PKSINNSDSESGNSFSLLMRRLGSIKTQRRTSIQKGYVQFFDIKTLEKATGGFKESSV
                                        IGQGGFGCVYKGCLDNNVKAAVKKIENVSQEAKREFQNEVDLLSKIHHSNVISLLGSA
                                        SEINSSFIVYELMEKGSLDEQLHGPSRGSALTWHMRMKIALDTARGLEYLHEHCRPPV
                                        IHRDLKSSNILLDSSFNAKISDFGLAVSLDEHGKNNIKLSGTLGYVAPEYLLDGKLTD
                                        KSDVYAFGVVLLELLLGRRPVEKLTPAQCQSLVTWAMPQLTDRSKLPNIVDAVIKDTM
                                        DLKHLYQVAAMAVLCVQPEPSYRPLITDVLHSLVPLVPVELGGTLRLTR"
                        }
                }
        }
        (ORIGIN,q):  "1 atgggaagtg gtgaagaaga tagatttgat gctcataaga aacttctgat tggtctcata
                        ...
                        1321 gtttcttttt attgtcaaat ataattatat ttttgt"
}
```

Figure 4. Representation of the RefSeq record NM_128079.4 (in Figure 3) using our quality-augmented semistructured data model.

3.2.3 Computing the Score of the Quality Measures

We are left now with the issue of how to effectively compute the score of each of our quality measures for a particular data item. We propose a set of heuristic formulae, which are largely based on historical information about the data (i.e., what was the value of the data in previous versions, how much time elapsed between versions, when was the last time it was updated, etc.). After analyzing some records from RefSeq together with their "revision history", we recognized the potential application of this historical data to the computation of our quality measures. Extracting the appropriate information from the different versions that a data item has undergone over time can reveal quality aspects such as stability, correctness, and currency of the data item at hand. We do not claim that our measure scores are optimal; in fact, there may even be more effective ways of computing the scores, so consider our formulae just as a starting point for future discussions on this subject.

For simplicity, most of the formulae we present here assume that the data graph is acyclic. However, they can be extended to account for cycles in the graph. In particular, we will assign scores to each of the quality measures in the quality metadata $q(l)$ of a label l as follows.

(i) If the label's target node $t(l)$ is an atomic value v, then:

- The **Stability** score Q_S in $q(l)$ is defined as

$$Q_S = 1 - \sum_{i=1}^{n} [\Delta(v(i-1), v(i)) \times \int_{t_i}^{t_{i-1}} \lambda e^{-\lambda t} dt] \qquad (1)$$

  where n is the number of versions v has undergone, t_i is the time elapsed since the ith version of v ($t_0 \equiv \infty$), and $v(i)$ is the state[2] of v at version i. Δ is a function that measures the proportion of change (if you want, the "distance") between two atomic values, which we will shortly define. The integral in formula (1) weights the changes depending on when they occur in time: recent changes have more weight than old changes; here $\lambda > 0$ is a parameter to be determined experimentally.

- The **Density** score Q_D in $q(l)$ is defined as

$$Q_D = 1 \qquad (2)$$

  which means that each atomic value counts as one data unit.

- The **TsLU** score Q_T in $q(l)$ is defined as

$$Q_T = \log\left(1 + \left\lfloor \frac{t-u}{f} \right\rfloor\right) \qquad (3)$$

  where t is the current time, u is the time when v was last updated, and f is the frequency of update[3] of the database. We use a log-based scale for time because we consider that it is fairer to convert time to a logarithmic scale when making comparisons.

- The **Redundancy** score Q_R in $q(l)$ is defined as

$$Q_R = \frac{|T_v| - 1}{|T_v|} \qquad (4)$$

  where T_v is the set of ssd-expressions in the database that are redundant with respect to v, and $|T_v|$ is the cardinality of this set. We assume that v itself is included in T_v so that if v is unique then $|T_v|=1$ and $Q_R=0$, and if v is redundant then $|T_v|>1$ and $Q_R>0$. In order to determine which data items are redundant one could use, for instance, the algorithms described in [11].

- The **Correctness** score Q_C in $q(l)$ is defined as

$$Q_C = w_1 \times Q_S + w_2 \times (1 - e^{-\beta \times age}) \qquad (5)$$

  where $\beta > 0$, $0 \le w_1 \le 1$, and $w_2 = 1\text{-}w_1$, are parameters to be determined experimentally. Q_S is the stability score, and *age* is the time elapsed since the creation of v. In other words, correctness is a weighted average of the stability and the age, with age being first mapped to the interval [0,1) (0 meaning that the data item is new, and a value close to 1 meaning that the data item is very old).

- The **Usefulness** score Q_U in $q(l)$ is defined as

$$Q_U = q(l).D \times q(l).C \times [1 - q(l).R] \qquad (6)$$

  where $q(l).D$, $q(l).C$, and $q(l).R$ are path expressions for the density, correctness, and redundancy scores associated to v.

We define the function $\Delta(v_1,v_2)$ for atomic values v_1 and v_2 as follows:

If v_1 and v_2 are of different type, then

$$\Delta(v_1, v_2) = 1$$

If v_1, v_2 are both strings, then

$$\Delta(v_1, v_2) = \frac{editDist\ (v_1, v_2)}{\max(length(v_1), length(v_2))}$$

If v_1, v_2 are both numbers, then

$$\Delta(v_1, v_2) = \frac{|v_2 - v_1|}{\max(v_1, v_2)}$$

(This formula assumes that v_1, v_2 are positive numbers.)

In any other case,

$$\Delta(v_1, v_2) = 1$$

Note that $0 \le \Delta(v_1, v_2) \le 1$ for any data pair (v_1, v_2). The aim of the Δ function is to assess the proportion of data that changed from value v_1 to value v_2. It can also be interpreted as a normalized distance between values v_1 and v_2. For example, if v_1 and v_2 are equal, then we need not change v_1 to obtain v_2, so $\Delta(v_1,v_2) = 0$. On the other hand, if v_1 and v_2 are strings and they differ in 50% of their content (i.e., 50% of v_1 needs to change so that v_1 becomes v_2), then $\Delta(v_1,v_2) = 0.5$.

[2] The state of an object is a given by its type and contents.

[3] Frequency of update indicates how often the database gets updated (e.g. daily, weekly).

(ii) If the label's target node $t(l)$ is a complex value $v = \{l_1 : e_1, \ldots, l_n : e_n\}$, then:

- The **Stability** score Q_S in $q(l)$ is defined as

$$Q_S = \frac{1}{n}\sum_{i=1}^{n} q(l_i).S \quad (7)$$

where n is the number of labels contained in v (i.e., outgoing edges from node $t(l)$), l_i is the i-th label in v, and $q(l_i).S$ is a path expression for the stability score associated to e_i (where e_i is a child node of $t(l)$).

- The **Density** score Q_D in $q(l)$ is defined as

$$Q_D = 1 + \sum_{i=1}^{n} q(l_i).D \quad (8)$$

where n and l_i are defined as in (7), and $q(l_i).D$ is a path expression for the density score of e_i. Q_D is in fact the number of nodes in the subtree whose root is v, i.e., the number of descendants of node v. We add one to this quantity to account for the node v itself. Formula (8) assumes that our data graph is effectively a tree (i.e., every node has only one parent node or incoming edge).

- The **TsLU** score Q_T in $q(l)$ is defined as

$$Q_T = \frac{1}{n}\sum_{i=1}^{n} q(l_i).T \quad (9)$$

where n and l_i are defined as in (7), and $q(l_i).T$ is a path expression for the TsLU score of e_i. Since the time for atomic values is given in a logarithmic scale, Q_T will not be too sensitive to extremely old data.

- The **Redundancy** score Q_R in $q(l)$ is defined as

$$Q_R = \frac{1}{n}\sum_{i=1}^{n} q(l_i).R \quad (10)$$

where n and l_i are defined as in (7), and $q(l_i).R$ is a path expression for the redundancy score of e_i.

- The **Correctness** score Q_C in $q(l)$ is defined as

$$Q_C = \frac{1}{n}\sum_{i=1}^{n} q(l_i).C \quad (11)$$

where n and l_i are defined as in (7), and $q(l_i).C$ is a path expression for the correctness score associated to e_i.

- The **Usefulness** score Q_U in $q(l)$ is defined as

$$Q_U = \frac{\sum_{i=1}^{n} q(l_i).D \times q(l_i).C \times \left[1 - q(l_i).R\right]}{\sum_{i=1}^{n} q(l_i).D} \quad (12)$$

where n and l_i are defined as in (7), and $q(l_i).D$, $q(l_i).C$, and $q(l_i).R$ are path expressions for the density, correctness, and redundancy scores of e_i, respectively.

At this point we can justify our formulae based on our initial experience with biological data, but we are in the process of validating this model with the development of a prototype and through posterior feedback from biologists.

3.3 Updating the Quality Measures under the Data Operations

Since we are primarily concerned about biological data, we cannot make the assumption that our data is mainly static. Conversely, we must consider a scenario where data is constantly updated through the operations defined in the data model. Thus, we need to address the issue of how our quality metadata is affected by each of the operations in the data model. For this purpose, we will consider a core set of operations on the graph model, which includes navigation, insertion, update, and deletion.

3.3.1 Navigating to a node n and returning its content

Let l be the last label (or edge) in the path through which we reached n, i.e., $t(l) = n$. None of the scores of the quality measures change under this operation. Together with the data represented by node n, this operation will return $q(l)$ as the quality metadata of the result.

3.3.2 Inserting a new node n

Let $l_1, l_2, \ldots, l_k$ be the path where n will be inserted, i.e. n will become a child of $t(l_k)$. Then, when the node is inserted we will assign initial scores to the quality measures of n. These initial scores are given by the formulae from previous section.

If the node to insert is a leaf (atomic value),

(a) Compute the quality measures for this leaf node according to formulas (1) through (6).

If the node to insert is an interior node (complex value),

(a) First, recursively compute the quality measures for its children (direct descendants).

(b) Then, compute the quality measures for this interior node according to formulas (7) through (12).

In either case, update the quality measures of all the nodes in the path from the root to n i.e., $t(l_1)$, $t(l_2)$,…, $t(l_k)$ so that they incorporate information about their new descendant. More specifically, this can be done using the procedure of an Update operation (see below).

Besides the path to the recently inserted node n, this operation will return the quality metadata $q(l_k)$ assigned to this new node.

3.3.3 Updating a node n

Let $l_1, l_2, \ldots, l_k$ be the path where n is located in the graph so that n is a child of $t(l_k)$. Then, when the node is updated, the scores of the quality measures will change according to the formulae from previous section.

If the node to update is a leaf (atomic value),

(a) Recompute the quality measures of this leaf node as specified by formulas (1) through (6).

If the node to update is an interior node (complex value),

(a) Recursively recompute the quality measures of those children that were affected by the update operation.

(b) Recompute the quality measures of this node as specified by formulas (7) through (12).

In either case, update the quality measures of all the nodes along the path from the root to n, i.e., $t(l_1)$, $t(l_2)$,…, $t(l_k)$.

This operation will return the updated quality metadata $q(l_k)$ of the just updated node n.

3.3.4 Deleting a node n

Let l_1, l_2, …, l_k be the path where n is located in the graph so that n is the child of $t(l_k)$. Then, when the node is deleted, the scores of the quality measures will be updated according to the formulae from previous section.

If the node to delete is a leaf (atomic value), then

(a) Delete label l_k (which contains the quality metadata for n).

(b) Recompute the quality measures of n's parent node, $t(l_{k-1})$, as specified by formulas (7) through (12).

If the node to delete is an interior node (complex value), we need to distinguish between two cases: (i) single node deletion, and (ii) subtree deletion. Hence,

(i) Single node delete (in this case, re-structuring of the tree is needed)

(a) Replace label l_k with the set of labels directly reached by l_k so that n's parent node becomes the parent of n's children nodes.

(b) Recompute the quality measures of n's parent node, $t(l_{k-1})$, as specified by formulas (7) through (12).

(ii) Subtree delete (no re-structuring of the tree)

(a) Delete label l_k (which contains the quality metadata for n) and all labels reachable from l_k in the subtree (which contain the quality metadata for the descendants of n).

(b) Update the quality measures of all the nodes in the path from the root to $t(l_{k-1})$ i.e., $t(e_1)$, $t(e_2)$, …, $t(e_{k-1})$) to account for the deletion of their descendants.

This operation will return the updated quality metadata $q(l_{k-1})$ of the just removed node's parent.

4. EXAMPLES

In this section, we illustrate the benefits of our model using several examples from biological databases.

Example 1. Suppose a biologist wants to have an estimate of the quality of the atomic value whose path expression is *Locus.Length-bp* (oid &9) in Figure 2. By using our quality-augmented data model, she obtains the following quality metadata for the atomic value 1356:

$$\{S: 0.9810, D: 1, T: 2.5366, R: 0, C: 0.9248, U: 0.9248\}$$

These scores were calculated using the twelve existing versions of the RefSeq record NM_128079 (as of March 2005), with parameters values $\lambda = 0.007$, $\beta = 0.002$, $w_1 = 0.4$ and $w_2 = 0.6$. The current time t in formula (4) was assumed to be date when the last version of the record occurred (January 25th, 2005), and f, the frequency of update of the database, was assumed to be 1 (daily updated). Also, we use a redundancy score of 0 for all our example values since RefSeq is by nature a non-redundant curated database.

The quality metadata shown above would be located at the edge (label) *Length-bp* in our model. Careful examination of these quality scores can provide greater insights into different quality aspects of the data at hand. For instance, we can immediately note that the current value of our atomic data was last updated 2.5366 log-time units ago (see *T* score). We can also know that this value is not redundant (see *R* score). Furthermore, we can deduce that this atomic value has had minimal or no changes during the most recent versions because its stability score is close to 1 (see *S* score). We can also be 92.48% confident about the accuracy of this atomic value (see *C* score). Based on the density score, we can infer that this atomic value conveys just one unit of data. The usefulness score suggests that 92.48% of the data in this atomic value is non-redundant and correct.

Figure 5 shows the behavior of the stability and correctness quality measures over time for the atomic value from Example 1. Time 0 corresponds to the date when this value was first inserted into the data source, and subsequent time markers correspond to the time elapsed since then, in years. Along with the quality scores, we also plot the magnitude of change from one version to the next one, which is defined by the Δ function. From Figure 5 we observe that whenever there is a change in the contents of our atomic value, the stability score either decreases or does not increase significantly with respect to its value on the previous version. Similarly, the correctness score is affected by changes to the value but it is not as sensitive to them as stability because it considers also the age of the data value. To exemplify this, observe how the scores for stability and correctness change from the version at time 1.6 to the version at time 2.05. The stability score drops from 0.91 to 0.79 due to a change of 27% in the contents of the value, whereas the correctness score remains at 0.78. This happens because the age of the value became significant, making correctness more robust to changes.

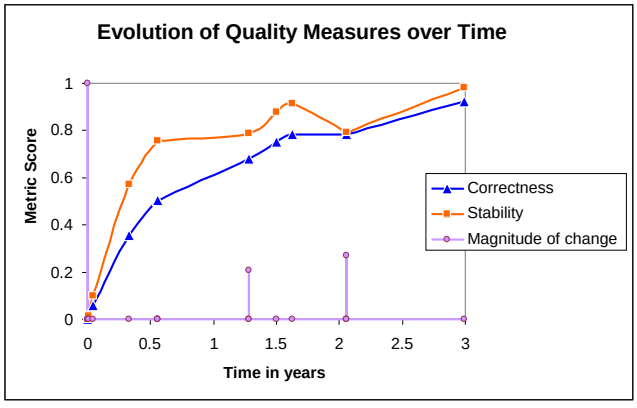


Figure 5. Evolution of Correctness and Stability scores over time for the atomic value at path *Locus.Length-bp*.

Example 2. Suppose the same biologist from Example 1 wants to have an estimate of the quality of the atomic value whose path expression is *Locus.Biomol* (oid &8) in Figure 2. By using our quality-augmented data model, she obtains the following quality metadata for the atomic value "mRNA":

$$\{S: 0.9995, D: 1, T: 3.0386, R: 0.5, C: 0.9322, U: 0.4661\}$$

These scores were calculated using the same parameters and settings as in Example 1. The quality metadata shown above is located at the edge (label) *Biomol* in our model. Let's examine these quality scores. We can see that the current contents of our atomic value were last updated 3.0386 log-time units ago, which suggest that this value has remained valid for longer time than the value from Example 1. We also note that this value is as dense as the value from Example 1 because each of them encloses one unit of data. It is also observed that the stability score is very close to 1, which implies that there were probably no significant changes to the contents of the value during the most recent versions. By comparing the stability score in this example and in Example 1, we deduce that this atomic value is currently more stable (i.e. has suffered less recent changes) than the value from Example 1. The usefulness score indicates that only 46.61% of the data in this atomic value conveys information (non-redundant correct data); thus assessing lower quality to this value than to the value in Example 1.

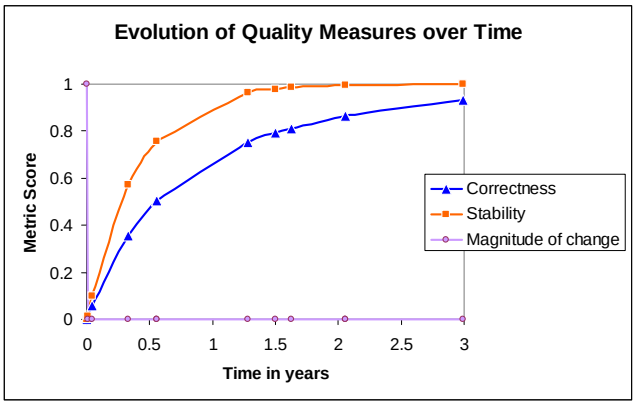


Figure 6. Evolution of Correctness and Stability scores over time for the atomic value at path *Locus.Biomol*.

Figure 6 shows the behavior of the correctness and stability quality measures over time for the atomic value *Locus.Biomol*. As before, time 0 corresponds to the date when this value was first inserted into the data source. Together with the quality scores, we plot the magnitude of change (defined by the Δ function) from one version to the next one. Note that the only change for this atomic value occurred when it was inserted. The curves for stability and correctness in Figure 6 show the typical convergence of these measures to their maximum values.

Example 3. Suppose our biologist from Examples 1 and 2 wants to have an estimate of the quality of the atomic value whose path expression is *Locus.Update-date* (oid &10) in Figure 2. By using our quality-augmented data model, she obtains the following quality metadata for the atomic value "25-JAN-2005":

$$\{S: 0.4676, D: 1, T: 0, R: 0.75, C: 0.7195, U: 0.1799\}$$

These scores were calculated using the twelve versions of the record NM_128079, and the same parameters from Example 1. The quality metadata shown above is located at the edge (label) *Update-date* in our model. The update-date value is not a very interesting data to analyze by itself since we know that it will change in every version and will simply contain the date when the last version took place. However, we use it here to illustrate the behavior of a value that changes frequently (as opposed to the data in Example 2, which does not change after insertion). Let's then examine the quality scores of this atomic value. The TsLU score of our atomic value is 0, which means that this value was updated during the last version and that the 'current time' is close to the time when this last version occurred. Note that the stability and correctness scores are significantly lower than the scores obtained in previous examples. In particular, a low stability score such as 0.4676 indicates the presence of large and recent changes in this atomic value (see Figure 7). The low correctness score is also an accumulated effect of the many changes experienced by this value over time, but it is especially influenced by the most recent changes. The correctness score of this atomic value is larger than the stability score because the age of the value is also considered. The usefulness score suggests that the fraction of non-redundant correct data conveyed by this atomic value is just 17.99% (lower than in Examples 1 and 2).

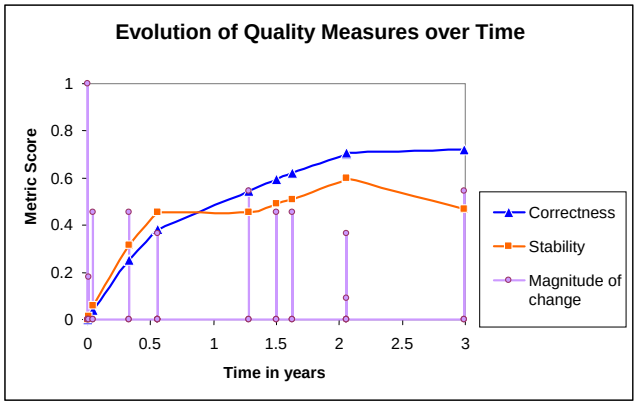


Figure 7. Evolution of Correctness and Stability scores over time for the atomic value at path *Locus.Update-date*.

Figure 7 shows the behavior of the correctness and stability quality measures over time for the atomic value *Locus.Update-date*. Observe that the relative change for this atomic value at each version is considerable (more than 0.3 or 30%). The stability and correctness scores from Figure 7 are far from reaching the maximum possible value (i.e., 1), and they will not increase much if this rate of change continues.

Example 4. Suppose our biologist from Examples 1, 2, and 3 needs to estimate the quality of the complex value whose path expression is *Locus* (oid &2) in Figure 2. By using our quality-augmented data model, she obtains the following quality metadata for the complex data at hand:

$$\{S: 0.8894, D: 6, T: 2.3305, R: 0.35, C: 0.8882, U: 0.6871\}$$

These scores were calculated using the same parameters as in Example 1. The quality metadata shown above is located at the edge (label) *Locus* in our model. We observe that the current

contents of our complex value were updated 2.3305 log-time units ago. We also note that 35% of the data in this complex value is redundant. The stability score of this complex value indicates that the average stability of its components is reasonably high. Likewise, the correctness score of this complex value suggests that the accuracy of its components, in average, is relatively high. A density score of 6 tells us that our complex value contain more data units (i.e., it is more dense) than the single atomic values from previous examples. We can infer from the usefulness score that 68.71% of the data contained in this complex value is accurate and non-redundant.

5. FUTURE WORK

We have proposed a quality-aware data model for representing and integrating quality metadata into a data source. Our model is tailored to biological data sources where ongoing concern over the low quality of rapidly growing experiment data has spurned the desire for quality assessment measures. We approached this problem in three steps.

First, we identified a set of quality measures relevant in a biological domain (but these measures could be easily extended to other domains). We give a clear and formal definition for each of the quality measures, specifying how to compute their score. Second, we selected a suitable data model and augmented it by incorporating the quality measures in a consistent manner. Third, we described how the quality measures are affected by each of the operations in the data model, and how the quality measures extend the result of the operations.

We plan on continuing the development of a prototype of our quality-aware model, which would allow us to validate the overall approach and demonstrate that the proposed quality measures are capable of providing meaningful and valuable information. An important part of the prototype is to implement the data operations in an efficient way so that updates are incremental. We also plan to explore ways for automatically capturing the data quality of existing data in biological repositories. This would enable a smooth migration from current to quality-aware data sources.

6. ACKNOWLEDGMENTS

Special thanks to Arturo Camacho for his useful discussions and reviews.

7. REFERENCES

[1] Abiteboul, S., Buneman P., Suciu, D. Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann Publishers, 2000.

[2] AGAVE - Architecture for Genomic Annotation, Visualization and Exchange. Available at http://www.agavexml.org/

[3] Ballou, D., Madnick, S., and Wang, R. Assuring Information Quality. *Journal of Management Information Systems*, *20*, 3(2004), 9-11.

[4] BSML -Bio Sequence Markup Language. Available at http://www.bsml.org/

[5] Buneman, P. Semistructured Data. *Proc. PODS '97.* Tucson, Arizona (May 1997).

[6] Buneman, S., Davison, S., Hillebrand, G., and Suciu, D. A query language and optimization techniques for unstructured data. *Proceedings of the ACM SIGMOD International Conference on Management of Data.* (1996), 505-516.

[7] Calvanese, D., De Giacomo, G., and Lenzerini, M. Modeling and Querying Semi-Structured Data. *Networking and Information Systems Journal*, *2*, 2(1999), 253-273.

[8] DDBJ -DNA Data Bank of Japan. Available at http://www.ddbj.nig.ac.jp/

[9] EMBL Nucleotide Sequence Database. Available at http://www.ebi.ac.uk/embl/

[10] GenBank. Available at http://www.ncbi.nlm.nih.gov/Genbank/index.html

[11] Hammer, J. and Pluempitiwiriyawej, C. Element matching across xml sources using a multi-strategy clustering technique. *Data and Knowledge Engineering (DKE), Elsevier Science*, *48* (2004), 297-333.

[12] Lee, Y.W. and Strong, D. M. Knowing-Why About Data Processes and Data Quality. *Journal of Management Information Systems, 20,* 3 (Winter 2003-4), 13-39.

[13] Lee, Y.W., Strong, D. M., Kahn, B.K., and Wang, R.Y. AIMQ: A methodology for information quality assessment. *Information & Management, 40,* 2(2002), 133-146.

[14] McHug, J., Abiteboul, S., Goldman, R., Quass, D., and Widom, J. Lore: A database management system for semistructured data. *SIGMOD Record*, *26*, 3(1997).

[15] Mecella, M., Scannapieco, M., Virgillito, A., Baldoni, R., Catarci, T., Batini, C. Managing Data Quality in Cooperative Information Systems. *Journal of Data Semantics*, *I* (2003), LNCS 2800.

[16] Mihaila, G., Raschid, L., Vidal, M. E. Querying "quality of data" metadata. *Proc. of the Third IEEE Meta-Data Conference.* Bethesda, Maryland (April 1999), 526-531.

[17] Missier, P., Batini, C. A Multidimensional Model for Information Quality in Cooperative Information Systems. *Proceedings of the Eighth International Conference on Information Quality* (2003), 25-40.

[18] Müller, H., Naumann, F., Freytag J.C. Data Quality in Genome Databases. *Proceedings of the Eighth International Conference on Information Quality* (2003), 269-284.

[19] Naumann, F., Freytag J.C., Leser, U. Completeness of integrated information sources. *Information Systems*, *29*, 7(2004), 583-615.

[20] NCBI Reference Sequences. Available at http://www.ncbi.nlm.nih.gov/RefSeq/

[21] Orr, K. Data Quality and Systems Theory. *Communications of the ACM, 41,* 2(1998), 66-71.

[22] Pipino, L.L., Lee, Y. W., and Wang, R. Y. Data Quality Assessment. *Communications of the ACM, 45,* 4(2002), 211-218.

[23] Scannapieco, M., Virgillito, A., Marchetti, M., Mecella, M., Baldoni, R. The DaQuinCIS Architecture: a Platform for Exchanging and Improving Data Quality in Cooperative

Information Systems. *Information Systems*, *29*, 7(2004), 551-582.

[24] Strong, D., Lee, Y., and Wang, R. Data quality in context. *Communications of the ACM, 40*, 5(1997), 103-110.

[25] The Biopolymer Markup Language –BIOML, Working Draft Proposal. Available at http://www.proteome.ca/x-bang/bioml/b_toc.htm

[26] Wand, Y. and Wang, R. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM, 39*, 11(1996), 86-95.

[27] Wang, R.Y., Reddy, M. P., and Kon, H.B. Toward quality data: An attribute-based approach. *Decision Support Systems, 13* (1995), 349-372.

[28] XEMBL. Available at http://www.ebi.ac.uk/xembl/

ETL Queues for Active Data Warehousing

Alexandros Karakasidis
Univ. of Ioannina
Ioannina, Hellas
alex@cs.uoi.gr

Panos Vassiliadis
Univ. of Ioannina
Ioannina, Hellas
pvassil@cs.uoi.gr

Evaggelia Pitoura
Univ. of Ioannina
Ioannina, Hellas
pitoura@cs.uoi.gr

ABSTRACT

Traditionally, the refreshment of data warehouses has been performed in an off-line fashion. Active Data Warehousing refers to a new trend where data warehouses are updated as frequently as possible, to accommodate the high demands of users for fresh data. In this paper, we propose a framework for the implementation of active data warehousing, with the following goals: (a) minimal changes in the software configuration of the source, (b) minimal overhead for the source due to the active nature of data propagation, (c) the possibility of smoothly regulating the overall configuration of the environment in a principled way. In our framework, we have implemented ETL activities over queue networks and employ queue theory for the prediction of the performance and the tuning of the operation of the overall refreshment process. Due to the performance overheads incurred, we explore different architectural choices for this task and discuss the issues that arise for each of them.

1. INTRODUCTION

The demand for fresh data in data warehouses has always been a strong desideratum from the part of the users. Traditionally, the refreshment of data warehouses has been performed in an off-line fashion. In such a data warehouse setting, data are extracted from the sources, transformed, cleaned and eventually loaded to the warehouse. This set of activities takes place during a loading window, usually during the night, to avoid overloading the source production systems with the extra workload of this workflow.

Still, users are pushing for higher levels of freshness. *Active Data Warehousing* refers to a new trend where data warehouses are updated as frequently as possible, due to the high demands of users for fresh data. The term is also encountered as 'real time warehousing' for that reason [22]. To give an example, we mention [3], where a case study for mobile network traffic data is discussed, involving around 30 data flows, 10 sources, and around 2TB of data, with 3 billion rows. The throughput of the (traditional) population system is 80M rows/hour, 100M rows/day, with a loading window of only 4 hours. The authors report that user requests indicated a need for data with freshness at most 2 hours.

This kind of request is technically challenging for various reasons. First, the source systems cannot be overloaded with the extra task of propagating data towards the warehouse. Second, it is not obvious how the active propagation of data can be implemented, especially in the presence of legacy production systems. The problem becomes worse since it is rather improbable that the software configuration of the source systems can be significantly modified to cope with the new task (both due to (a) the down-time for deployment and testing and (b) the cost to administrate, maintain and monitor the execution of the new environment).

So far, research has mostly dealt with the problem of maintaining the warehouse in its traditional setup [10, 15, 16, 18, 21]. Related literature presents tools and algorithms for the population of the warehouse in an off-line fashion. In a different line of research, data streams [1, 5, 17] could possibly appear as a potential solution. Nevertheless, at least until now, research in data streams has focused on topics concerning the front-end, such as on-the-fly computation of queries, without a systematic treatment of the issues raised at the back-end of a data warehouse. For example, to our knowledge, there is no work related to how streaming data are produced or extracted from data producers; not to mention the extra problems incurred when the data producers are operational systems.

To this end, in this paper we attempt to approach the problem from a clean sheet of paper. We investigate the case where the source of the warehouse is a legacy system. The specific problem involves the identification of a software architecture along with appropriate design guidelines for the implementation of active warehousing. We are motivated by the following *requirements* in achieving this goal.

1. *Maximum freshness of data*. We want to implement an active data warehousing environment to obtain as fresh data as possible in the warehouse.
2. *Smooth upgrade of the software at the source*. We wish to implement a framework where the modification of the software configuration at the source side is minimal.
3. *Minimal overhead of the source system*. It is imperative to impose the minimum additional workload to the source.
4. *Stable interface at the warehouse side*. It would be convenient if the warehouse would export a stable interface for its refreshment to all its source sites.

The grand view of our environmental setup is depicted in Figure 1. A set of *sources* comprise source data and possibly *source applications* that manage them (for the case of legacy sources) or DBMS's for the case of conventional environments. The changes that take place at the sources have to be propagated towards the warehouse. Due to reasons of semantic or structural incompatibilities, an *intermediate processing stage* has to take place, in order to transform and clean the data. We refer to this part of the system as the *Active Data Staging Area* (ADSA). Once ready for loading, the data from the intermediate layer are loaded at the warehouse, through a set of *on-line loaders*.

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

Mapping this grand view to concrete technical choices requires the tuning of several components of the architecture. Following, we quickly summarize our findings that affected our architectural choices.

Starting with the sources, in this paper, we have focused on legacy systems. Apart from the requirement of minimal changes at the source side, legacy sources pose the interesting problem of having an application (instead of a DBMS) managing the data. We modify a library of routines for the management of data to allow the interception of the calls without affecting the applications. The modification involves (a) inserting no more than 100 lines of code to a library of routines for source management and (b) recompiling the application (which was not affected), over this library. Also, as far as the communication between stages is concerned, we transmit blocks of records for reasons of performance and minimal overhead of the source system.

The internal architecture of the intermediate layer (ADSA) is not obvious, either. For each ETL activity, we employ a queue to store incoming records before they are processed. Each activity processes the incoming data on-line and then passes its output to the next queue for further processing. Again, for reasons of performance, the unit of exchange is blocks of records and not individual records. We do not assume a fixed set of ETL operators, but rather we provide a taxonomy of such operations, based on their operational semantics. New operators can be added to the taxonomy as they are defined. To predict the performance of the system, we employ queue theory for networks of queues (cf. section 2.1 for a reminder of queue theory). Our experimental results indicate that the assumption of an M/M/1 queue for each of the ETL activities provides a successful estimation.

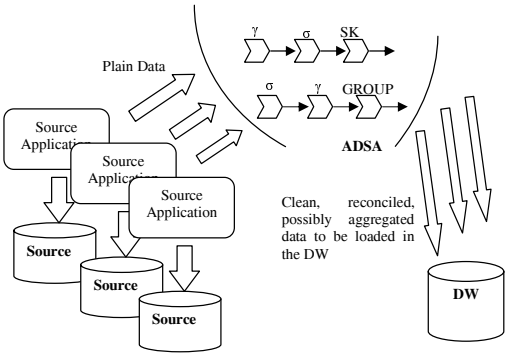


Fig. 1. Architecture Overview

To implement the requirement for stable interface at the side of the warehouse, the data are further propagated towards the warehouse through an interface involving web services [2]. The need for web services as the technical solution for populating the warehouse with fresh data is not self-evident and requires justification. In fact, web services are known to be rather heavy middleware in terms of resource consumption [9], which potentially jeopardizes the requirement of fresh data and minimal overhead. The main advantages of web services compared to other middleware solutions (RPC, ORB's, message queues, etc) are: (a) interoperability, meaning that they can be deployed in all platforms and configurations and (b) possibility of exporting them outside the intranet of an organization. We emphasize the interoperability property: in a large organization, there is a wide variety of data sources, involving several platforms and configurations. Web services are syntactically reliable, as they can provide a common, stable interface for the warehouse to all these sources without requiring major design and integration effort. Also, this loose coupling of sources and the warehouse results in minimal impact in the case of changes, either at the source or at the warehouse. Obviously, performance has been a concern too. Still, as we discuss in Section 4, our experiments indicate that the overall delay, incurred by the adaptation of a solution based on web services is rather small, especially if one is willing to trade resources (mainly main memory) for freshness.

In a nutshell, our contributions can be listed as follows:

- We set up the architectural framework and the issues that arise for the case of active data warehousing.
- We develop the theoretical framework for the problem, by employing queue theory for the prediction of the performance of the system. We provide a taxonomy for ETL tasks that allows treating them as black-box tasks, without the need of resorting to algebraic, white-box descriptions of their functionality. Then, standard queue theory techniques can be applied for the design of an ETL workflow.
- We provide technical solutions for the implementation of our reference architecture, achieving (a) minimal source overhead, (b) smooth evolution of the software configuration at the source side and (c) fine-tuning guidelines for the technical issues that appear.
- We validate our results through extensive experimentation. Our implementation suggests that our theoretical formulation successfully predicts the actual performance of the system.

The rest of this paper is organized as follows. In Section 2, we set up the problem theoretically and in Section 3, we present the different architectural choices and the technical challenges that each of them incurs. In Section 4, we present the experimental evaluation of the proposed framework. Finally, in Section 5, we present related work and in Section 6, we conclude with our results and present topics for future research.

2. QUEUE THEORY FOR ETL ACTIVITIES

In our architecture, data flows from the sources towards the warehouse, through an intermediate data processing stage. In this stage, data sustain various types of filtering and transformations. We employ queue theory as the cost model that predicts the data delay and the system overhead at this intermediate stage. We model each ETL activity as a queue in a queuing network. We provide a simple taxonomy for ETL activities, showing how to derive a simple queue model for them, without delving into their internal semantics. In this section, we start with some fundamentals of queue theory, then we move on to discuss a taxonomy of ETL operations and, finally, we conclude with the presentation of queue networks.

2.1. Preliminaries

Fundamentally, in a queuing model, a sequence of customers arrives at a server. If a customer arriving at the server finds the server occupied, it waits in the queue until its turn to be served comes. After the customer is served, it leaves the system [11]. If λ customers arrive at the system per time unit, then the mean inter-

arrival time is equal to $1/\lambda$. Similarly, if μ customers leave the system per time unit, then the mean service time is equal to $1/\mu$. Based on these parameters, we also define $\rho=\lambda/\mu$ as the traffic intensity which denotes the server utilization. We require that $\rho<1$ or the queue length can become unbounded.

The distribution of the arrival and the service rates can take different values (Poisson, constant, etc). Depending on these distributions, different equations hold for predicting the mean length of the queue and the mean service time for each customer. A full discussion of these properties falls outside the scope of this paper; therefore we refer the interested reader to [11, 23] for a detailed discussion.

A fundamental relation between the mean number of customers in the system N, the customer mean arrival rate in the system λ, and the mean time T that a customer remains in the system is given by Little's law. This relation is formulated as $N=\lambda*T$ and its importance resides in the fact that this equation holds for every type of queuing system irrespectively of the arrival and service rate distributions. By applying Markov Theory and Little's law to a queue with Poisson arrivals and exponentially distributed processing times, (also known as M/M/1 queue), we can estimate the mean response time of the system $W = 1/(\mu—\lambda)$ and the mean queue length $L=\rho/(1-\rho)$.

2.2. A Taxonomy of ETL Activities

Each ETL queue can direct customers to more than one subsequent queue, depending on the type of operation it performs. In queue theory, the composition of queues is treated by queue networks. The computation of the interesting properties of such networks depends on the nature of the involved individual queues. The question that arises is what kind of individual queues do the ETL activities produce. One possible way to answer this question is to define an extension of the relational algebra, specifically tailored for ETL purposes and study the properties of each operator from the viewpoint of queue theory. Since this would probably produce quite complex queues, we adopt a different, black-box approach and define a taxonomy of ETL transformations, based on the relationship of their input and output. This way, we practically categorize ETL tasks in families without delving in the particularities of their internal functionality. Specifically, the taxonomy of activities consists of the following categories: (a) Filters, (b) Transformers and (c) Binary Operations.

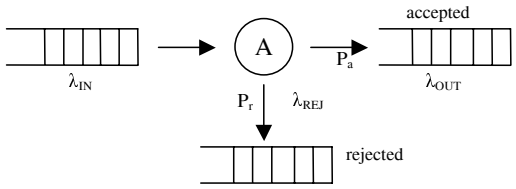


Fig. 2. Queuing model for multi-output activities

Filters examine each incoming tuple to determine whether it meets certain criteria. If these criteria are fulfilled, then a tuple is accepted and propagated towards an acceptance output. If not, it is rejected and possibly propagated towards a rejection output. We assume that tuple arrivals occur due to a Poisson process and service times follow an exponential distribution. We define the probability that some tuple i is accepted as P_a and the probability that some tuple i is rejected by the system as P_r. This is illustrated in Figure 2. It is obvious that $P_a+P_r=1$.

The filtering operations do not impose a change to the overall number of tuples making the following equation valid:

$$| \textit{tuples entering service} | = | \textit{tuples accepted} | + | \textit{tuples rejected} |$$

Also, these operations do not incur changes to the schema of the tuples entering the service facility compared to the schema of the exiting tuples. Typical operations of this category are not-null, domain and foreign key checks, selections, and in general, any type of operation, operating locally on a tuple and determining whether it will be further propagated or not. Due to their multiple outputs, filters can also act as routers for tuples whose destination depends on their value.

Considering the case of **Transformers**, tuples entering a transformer undergo changes to their value and/or their schema. We can distinguish two subclasses of Transformers taking into account the relationship between the number of tuples entering and the number of tuples exiting the transformation.

In the first case the two quantities are equal which means:

$$| \textit{tuples entering service} | = | \textit{tuples accepted} |$$

We assume that tuple arrivals occur due to a Poisson process and service times follow an exponential distribution, in other words we have the same case with filters transformations. Again, we define the probability that some tuple i is accepted as P_a and the probability that some tuple i is rejected by the system as P_r. Since all tuples are accepted, we have: $P_a=1$ and $P_r=0$ (Figure 3). Examples of such transformations are the surrogate key transformation, the usage of functions for the derivation of new values and, in general, any transformation that derives an output tuple solely on the basis of the value of a single input tuple.

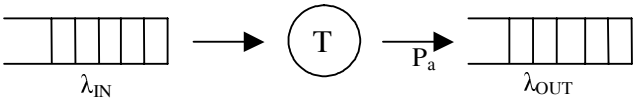


Fig. 3. Queuing model for single-output activities

In the second case, the number of tuples entering the system is different compared to the number of tuples exiting and in specific:

$$| \textit{tuples entering service} | > | \textit{tuples accepted} |$$

This occurs because some of the tuples entering service are aggregated or merged. We assume that tuple arrivals occur due to a Poisson process and service times follow an exponential distribution. The problem with this kind of transformations is that practically queue customers disappear and new customers are produced by each transformation. To model this property in terms of queue theory, we make the assumption that depending on the aggregation or merging factor, some of the incoming customers continue and some exit the system. In other words, we assume that some of the tuples, after being transformed, continue through the system as accepted. The number of these tuples equals the number of tuples produced as a result of the transformation. The rest of the tuples are assumed to be rejected by the system after their service and exit the system. The following equation holds:

$$| \textit{tuples rejected} | = | \textit{tuples entering service} | - | \textit{tuples accepted} |$$

Again, we define as P_a the probability that some tuple i is accepted and P_r the probability that some tuple i is rejected by the system: $P_a+P_r=1$. Given the aggregation factor of an incoming set of data, we can easily compute the acceptance and rejection rate as well as the respective routing probabilities. The routing probabilities are:

$$P_a = \frac{|result\_tuples|}{|input\_tuples|} \text{ and } P_r = \frac{|input\_tuples| - |result\_tuples|}{|input\_tuples|}$$

The third class of ETL activities deals with **Binary** operators. This is the case where data from multiple sources are combined and a single outgoing stream is produced. Examples of such operations involve variants of the join operation, including the join of data from different tables, as well as difference and update detection operations among different snapshots of the same table. [14] describe a window-based hash join algorithm for continuous streams. In the context of ETL, we make the following assumptions and observations:

- One of the two inputs is considered as the *primary input flow*. Tuples of this flow are checked over filters or transformed according to the values of some other relation and ultimately, either propagated towards the warehouse or rejected.
- The second input of the operator is acting as a *regulator of the primary flow*. In other words, its values are only needed in order to determine the processing and routing of the tuples of the primary flow. For all practical purposes where active ETL functionality is needed (update detection, difference, facts joined with dimension values), a static snapshot of the regulator flow can even be assumed.
- Adopting the model of [14], both inputs arrive at the same queue – they simply undergo processing with different distributions of processing times.

In principle, a binary operator has to be dealt with as a multi-class queuing system, with one class for each flow (input or output) – see Figure 4. We refer the interested reader to [14] for such a treatment. Still, based on the aforementioned assumptions, we can avoid modeling the system as a multi-class queue, and deal only with the primary flow of the operator. In the rest of the paper, we will consider single-class queues, the tuples of which either (a) continue in the system or (b) are ultimately rejected. An interesting observation here is that no matter how many different categories of tuples enter the node for service, the output tuples can be assumed to belong in one of the two aforementioned categories.

We consider Poisson arrivals and exponential service times. As stated earlier, the two routing classes are accepted with probability P_a and rejected with probability P_r and as before $P_a+ P_r=1$. This type of operations does not impose a change to the overall number of tuples existing making the following equation valid (Figure 4):

$$|\ tuples\ entering\ service\ | = |\ tuples\ accepted\ | + |\ tuples\ rejected\ |$$

However, differently from Filters, the schema of the tuples possibly changes.

We can generalize the three aforementioned classes, through a **Generic Model**, where a node consisting of a single server serves possibly more than one classes of customers. All customers arrive according to a Poisson process and are serviced with exponential service times. The general case is depicted in Figure 4.

In the general case we can assume that tuples belonging to one of the two different classes of customers, say c_i, after their ETL transformation at the node, leave the system with probability P_{ri} and continue in the queue network with probability P_{ai}. Concerning the number of tuples in the system the following equation is still valid:

$$|\ tuples\ entering\ service\ | = |\ tuples\ accepted\ | + |\ tuples\ rejected\ |$$

Concerning the schema of the tuples before and after service, we observe that the schema changes in the general case, apart from the case of filters.

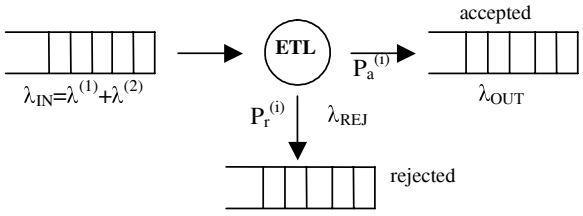


Fig. 4. Generic Model for ETL Queues

In the rest of this paper, we will follow the assumption of a primary input flow. This obviously results in forming an M/M/1 queuing node as the constructing element of our ETL queue network.

2.3. Queue Networks for ETL queues

Many queuing systems consist of a network of queues. In a *queuing network* (QN), a customer finishing service in a service facility is either immediately proceeding to another service facility or leaves the system. For our purposes, we assume that each node of this network consists of a single server with exponential arrival and exponential service times. One basic classification of queuing networks is the distinction between open and closed queuing networks. In an open network, new customers may arrive from outside (coming from a conceptually infinite population) and later on leave the system. In a closed queuing network, the number of customers is fixed and no customer enters or leaves the system. In our case, we are exclusively interested in open networks.

If an open queuing network is in steady state (i.e., the number of customers in the queue has converged over time), then for each node i, its arrival rate λ_i equals its departure rate μ_i. The arrival rate λ_i to node i is clearly the sum of all arrivals to i (including i itself). Assuming that i has N neighbors, the rate of external arrivals is λ_{0i}, and the probability of an arrival from its j-th neighbor is $p_{j,i}$, we have: $\lambda_i = \lambda_{0i} + \sum_{j=1}^{N} p_{j,i}\lambda_j$

These equations are called traffic equations and they can be transformed into a set of N simultaneous linear equations with a unique solution for M/M/1 nodes. In order to calculate the performance measures in queuing networks the steady state probabilities $\pi(k_1,\ldots, k_N)$ have to be found. The term $\pi(k_1,\ldots, k_N)$ denotes the probability of k_1 customers in queue 1, k_2 customers in queue 2 and so on. To this end, we employ Jackson's theorem that allows the calculation of the steady state probabilities of the whole network by separately calculating the probabilities of each

node, under reasonable assumptions (that our ETL queues fulfill) [23].

Jackson's Theorem [23]. If in an open network the condition $\lambda_i < \mu_i \cdot m_i$ holds for every $i \in \{1, ..,N\}$ (with m_i standing for the number of servers at node i) then the steady state probability of the network can be expressed as the product of the state probabilities of the individual nodes:

$$\pi(k_1,..., k_N) = \pi_1(k_1)\pi_2(k_2)... \pi_N(k_N)$$

Therefore, we can solve this class of networks in four steps:

1. Solve the traffic equations to find λ_i for each queuing node i.
2. Determine separately for each queuing system i its steady-state probabilities $\pi_i(k_i)$.
3. Determine the global steady-state probabilities $\pi(k_1,..., k_N)$. Derive the desired global performance measures.
4. From step 1, we can derive the mean delay and queue length for each node.

Methodology. How can we exploit the aforementioned theoretical analysis for designing ETL workflows for active data warehousing? *The design problem for active data warehousing involves predicting the mean delay and the queue length of ETL queues in the ADSA, given the source production rates and the processing power of the ADSA and the DW.*

The methodology for this task is straightforward. First, we classify each ETL task that we need to perform in one of the categories of our taxonomy. Then, we construct a queue network of such ETL queues. Finally, we solve the network equations as mentioned above.

3. FRAMEWORK AND ISSUES RAISED

Apart from the theoretical issues, there are several issues concerning the implementation of an active data warehouse. Therefore, in this section, we will start by presenting the general architecture of such a system. In subsection 3.1, we present the grand view for active warehousing and its specific instantiation that we have investigated. Then, in subsection 3.2, we proceed to a detailed presentation of the issues raised within this framework.

3.1. System Architecture

Our architecture consists of the following elements: a Data Source generating data, an intermediate data staging area that will be referred to as the Active Data Staging Area (ADSA) where the processing of data takes place and the Data Warehouse (DW). The architecture is illustrated in Figure 5.

The source comprises a data store (legacy or conventional) and an operational data management system (e.g., a DBMS or an application, respectively). Changes that take place at the source side have to be propagated towards the warehouse, which typically resides in a different host computer. The communication between hosts employs a network protocol (e.g., TCP or UDP). To avoid the extra overhead of overloading the network with half-full packets and, as our experiments indicate, to avoid overloading the source with the extra task of performing this task, we employ a *Source Flow Regulator* (SFlowR) module that compiles changes in blocks and propagates them towards the warehouse.

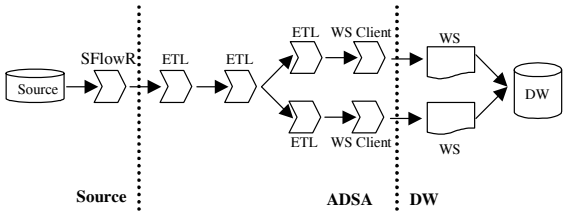


Fig. 5. Architecture Overview

Once record blocks have left the source, an ETL workflow receives them at the intermediate staging area. The role of the ETL workflow is to cleanse and transform the data in the format of the data warehouse. The ETL workflow comprises a set of ETL activities, also called *ETL queues*, each pipelining blocks of tuples to its subsequent activities, once its filtering or transformation processing is completed. In order to perform this task, each ETL activity checks its queue (e.g., in a periodic fashion) to see whether data wait to be processed. Then, it picks a specified number of records, performs the processing and forwards them to the next stage. If less than the specified records exist in the queue, then they are all retrieved. If the queue is empty, then the invocation is postponed, until there exist data to be processed.

The role of the active data staging area is versatile: (a) it performs all the necessary cleansings and transformations, (b) it relieves the source from having to perform these tasks, (c) it can act as a regulator for the data warehouse, too (in case the warehouse cannot handle the online traffic generated by the source) and (d) it can perform various tasks such as checkpointing, summary preparation, and quality of service management.

Once all ETL processing is over, data are ready to be loaded at the warehouse. As already explained, we chose to perform this task through a heavy but reliable (syntactically and operationally) middleware, web services. For each target table or materialized view at the warehouse, we define a receiving web service. To be able to invoke the web service, a client needs to be constructed. To regulate the traffic between the staging area and the warehouse, the client compiles the data in blocks, too. The web service at the warehouse side then populates the target table it serves. Load-balancing mechanisms at the warehouse side and physical warehouse maintenance (e.g., index maintenance) can also be part of this architecture. Still, for the moment, we do not address these problems.

In terms of the particular implementation that we examine in this paper, we have studied the problem as it appears over legacy sources. In our configuration, the source includes two software modules: (a) an ISAM file and (b) an application used to modify data in the legacy data source. In order to manipulate ISAM files, there is a library of ISAM routines that are invoked from the application at the source side. We have modified these library routines in order to replicate the data manipulation commands and send updates towards the staging area. Several ETL queues reside at the staging area performing cleanings, transformations and aggregations. Each ETL activity retrieves data from its queue with a constant rate, retrieving a given number of elements in constant intervals. ETL activities communicate both with each other and with the web service clients via Java thread-safe queues. The transfer from the staging area towards the Data Warehouse is done over HTTP (implying TCP as the underlying network protocol).

For our experiments, we assume that the warehouse simply stores the data performing no other task.

3.2. Issues Raised

In order to fulfill all the goals mentioned in Section 1, using the architectural elements described above, there are some issues raised which mainly concern the tuning and configuration of the system. The key issues that affect system performance and need to be resolved are discussed in this section and classified with respect to their locality at the source or the staging area, as well as the overall setup of the environment. All the technical choices and their alternatives are summarized in Table 1.

3.2.1 Choices concerning the Topology

Having described our architectural elements, the next step is to determine their topology. Our architecture offers the ability of selecting different number of tiers. Several choices exist:

- Two-tier architecture, where the source and the warehouse are found on different machines. There are two alternatives concerning this choice: the first is to place the staging area together with the source, putting the data warehouse on a separate machine. The second alternative is to place the staging area at the host where the data warehouse resides (Figure 6).
- Three-tier architecture, where we use a separate dedicated machine for the staging area, leading to a three-tier topology.

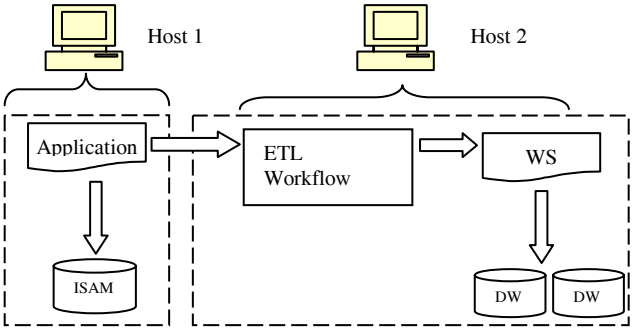


Fig. 6 Two-tier topology: The Data Warehouse and the ADSA reside on the same host, while the Source resides on a separate machine.

Coming to the two-tier architecture, the main issue that arises is related to the placement of the staging area. In the case of the staging area placed at the source, data warehousing operations do not burden the source, but still the resources used by the web services API to perform the invocation remain considerable. A way for dealing with this is to move the staging area to the warehouse host (Figure 6), which can be expected to be more powerful from the source host. This way, the source is completely detached from the active data warehousing process. Naturally, if the warehouse server is too loaded or its configuration too complex for the extra software setup of a web service server, the three-tier architecture can also be employed. Using the three tier architecture solves all the abovementioned problems, but increases the setup and maintenance cost, since an extra server, apart from the one used from the warehouse, has to be engaged and administered.

Having discussed the architectural alternatives for our topology, we can now proceed to discuss the technical issues raised for each of the main components and their overall setup.

3.2.2 Choices concerning the Source

Concerning the source side, the first consideration that arises has to do with the interconnection type between the source and the staging area. Since our goals are to impose as little impact as possible to the source and to make only minor changes, we have chosen the solution of sockets both due to its anticipated (but not thoroughly tested) lighter footprint characteristics and the easiness of programming such a solution.

The next choice is between TCP and UDP protocols for the transmission of data between the source and the staging area. On one hand, TCP offers reliability. On the other hand, UDP offers speed through non-blocking calls, followed by a concern on the server side for the socket buffer size, in case of extended datagram bursts and no reliability.

A third architectural choice concerns the way that changes to the source file are written to the socket, i.e., whether data are organized in blocks before being further propagated to the staging area. There are two ways to deal with this issue: either to write each modification to the socket, or to write bulks of modification commands. In the first case, whenever a data manipulation command is issued, it is immediately written to the socket along with the respective data. In the second case, nothing is written, until a number of records is completed. Then, all records together are sent to the staging area.

3.2.3 Choices concerning the Staging Area

The internal structure of the data staging area and the tuning of its operation are the major issues concerning the performance of our architecture. The staging area is a multithreaded environment with shared components, thus having to be set up properly to avoid race conditions and consistency.

The problem of locking raises the issue of queue emptying rate. Assuming that the input to the staging area is determined by the workload of the source (i.e., it cannot be constrained by the warehouse administrator), a proper emptying rate for the ETL queues has to be determined. A high arrival rate compared to the configured service rate will result in instability and queue length explosion. On the contrary, a very high service rate potentially results in too many locks of the queue (resulting again in delay, contrary to what would normally be expected). It is obvious that the service rate should be close to the arrival rate in order to have both efficient service times, and as less locks as possible.

Another dilemma is related to the interconnection type between the staging area and the data warehouse. As already mentioned, the staging area invokes a web service residing at the warehouse side. Although the SOAP protocol is one-way and asynchronous, implementations abide by the traditional middleware conventions of remote invocation, namely (a) *blocking* and (b) *non-blocking*. Blocking invocation involves an acknowledgment message to be sent from the web service, before its client can continue. In our case, this means that a response from the warehouse is required, delaying however the queue emptying rate. Non-blocking invocation does not delay the queue-emptying process of the web service client, since no response is returned from the invocation.

Finally, the issue of sending data as tuple-at-a-time or blocks is raised again for the communication between the staging area and the warehouse. In this case, apart from the network overhead, the cost of parsing the incoming web service messages at the warehouse plays a role for this choice.

3.2.4 Choices concerning the Warehouse

The data warehouse side is characterized by a web wervice per target table, receiving the cleansed data from the data staging area. The web services API offers three ways of handling the remote invocations of the client that resides in the data staging area. The first way is to create a single web service instance that handles all incoming requests. The second way is to create an instance for every session, and the third is to create an instance for each invocation request. In our configurations, we use the first of these alternatives. The reason is that in our experiments, we have employed one client for the service, which stops its operation after inserting a specific amount of records into the ISAM file. This makes the case of using an instance per session the same as using a single instance. Using an object per request is prohibitive, since we assume high frequency invocations.

Table 1. Architectural choices

Issue	Alternatives
General Architecture	
Topology	- 2-tier, ADSA at the source side - 2-tier, ADSA at the DW side - 3 tier
Source	
Connection Type	- UDP - TCP
Propagation Type	- One at a time - Block-based
Active Data Staging Area	
Interface between the two APIs	- None - Synchronized Queue
Web Service invocation type	- Blocking - Non Blocking
Propagation Type	- One at a time - Block-based
Data Warehouse	
Session management	- Single WS - Instance per session - Instance per request

4. EXPERIMENTS

In this section, we present the experiments we conducted. We present two sets of experiments. The first set presented in section 4.1 deals with the general behavior of the system. The purpose of this set of experiments is to figure out the behavior of each system component separately, and to establish guidelines for building the system. In this case, data are just transferred to the warehouse and no ETL operations are involved. In the second set of experiments, presented in section 4.2, we evaluate the behavior of our system in a realistic setup, based on the conclusions derived from the first set. Naturally, in this case, we also transform data using ETL operations.

Our experimental setup, which stands for both cases, is as follows: The ISAM library that we altered is the PBL/ISAM suite [20] available under GPL license. We have used a sample program distributed within the suite as the legacy application. We use two different data sets for our purposes. The first consists of 100,000 records and the second of 1,000,000 records. The ETL queues of the ADSA have been implemented using the Sun JDK 1.4, whose runtime engine has also been used. As a Web Services platform we have used Apache Axis 1.1 [4] with Xerces XML parser running over Apache Tomcat 1.3.29. Our data warehouse is implemented as a MySQL 4.1 database.

The host we used for the source was a PIII 700MHz with 256MB of physical memory running SuSE Linux 8.1. The host used as the data warehouse was a Pentium 4 2.8GHz with 1GB of physical memory running Mandrake Linux. This server also hosted the staging area. The hosts are interconnected via the switched Fast Ethernet LAN of our department.

Our data were created from the TPC-H data generation tool. For the first case, each row of data has fixed size equal to 20 bytes. In the second case, where we evaluate the system behavior under operational conditions, we used data of variable size. In this case each row has an average size of 140 bytes.

In our experiments we evaluate the cost in marginal conditions. Thus in order to evaluate the worst case, the source stores data at its peak capability. Moreover, since our warehouse host is a much faster computer than the source host, we would not be able to make safe conclusions if we let it operate at full capability (see also subsection 4.1.4). Thus we simulate slower server performance by employing timeouts between operations. This will be explained in more detail later.

4.1. Experiments on Architecture without ETL Processing

This section includes the first set of experiments we conducted. The aim of these experiments is to decide on basic architectural choices of our system. Throughout the experiments, the software operating at the staging area is a simple queue, called *Data Warehouse Flow Regulator* (DWFlowR), receiving source blocks of records and passing them to the warehouse.

4.1.1 Smooth Upgrade

One of the goals of our architecture is to pose minimal modifications to the source's code. In our approach, we do not alter the legacy application itself, but the library that manipulates the ISAM files by adding few lines of code to the routines that are of interest to the purpose of active warehousing. These routines are: the file opening routine, the record insertion routine and the file closing routine. The alterations are located only in the following four points of the library's source code:

1. The first modification is to include our library which contains the socket's client and the SFlowR.
2. The second modification is to add a call to the routine of our library that opens a socket to the staging area at the ISAM file opening routine. This call is performed only if the opening of the ISAM file is successful.
3. The third modification is to extend the insertion routine of the ISAM file library that writes the record to the file with a call to our library's function that propagates the change to the socket. This routine stores the specific record to the

SFlowR's buffer and when the defined number of records is completed, it delivers them to the staging area. Again, this routine is called only after a successful insertion.

4. The fourth modification is to add a call to the routine of our library that closes the opened socket to the staging area, at the ISAM file closing routine. This call is performed only if the closing of the ISAM file is successful.

Figure 7 shows the alterations that we have performed to the library in pseudo-code. The overall length of code that had to be written for this part of the implementation, including the additions at the ISAM library, is approximately 100 lines.

The routine that opens the socket to the DWFlowR reads configuration information from a plain text file, before the opening of the socket. This file contains the following three pieces of information:

1. The number of records the SFlowR will gather
2. The address of the DWFlowR
3. The port of the DWFlowR

Original Routine	Altered Routine

`Open_isam_File()`
`{`
`  …`
`opening_isam_file_co`
`mmands`
`  …`
`}`
	`Open_isam_File()`
	`{`
	`  …`
	`opening_isam_file_comm`
	`ands`
	`  …`
	`if(open==success)`
	`DWFlowR_socket_open()`
	`}`

`Write_record_to_File`
`()`
`{`
`  …`
`insert_record_comman`
`ds`
`  …`
`}`
	`Write_record_to_File()`
	`{`
	`  …`
	`insert_record_commands`
	`  …`
	`if(write==success)`
	`write_to_SFlowR()`
	`}`

`Close_isam_File()`
`{`
`  …`
`closing_isam_file_co`
`mmands`
`  …`
`}`
	`Close_isam_File()`
	`{`
	`  …`
	`closing_isam_file_comm`
	`ands`
	`  …`
	`if(close==success)`
	`DWFlowR_socket_close()`
	`}`

Fig. 7. Code alterations at the routine opening the ISAM file.

As an overall assessment of the impact of our changes, we can say that (a) minimal code had to be written to achieve the replication of incoming updates to the warehouse in an active fashion, (b) simple configuration parameters are required, (c) no changes were required to the code, rather than a simple recompilation under the new library.

4.1.2 UDP vs. TCP

The first parameter that needed to be tested involved the network protocol between the source and the staging area. The goal of our first experiment is to determine the system's behavior using UDP and specifically if there are any datagram losses. The results show a 35% packet loss of data, most probably due to the overflowing of data. Such losses are prohibitive for normal operation of an on-line environment. Therefore, for the rest of the paper, we have fixed TCP as the interconnection protocol between the source and the staging area.

4.1.3 Overhead at the Source

The main requirement for the architecture at the source side involves minimal overhead during regular operation. Therefore, the goal of the next experiment is to measure the overhead that our configuration incurs at the source side. We measure the time to complete the insertion of (a) 100 000 and (b) 1 000 000 to the ISAM file.

First, we measure the effect of using the SFlowR at the source. We try three values: 1, 100, and 1000 records for each packet that the SFlowR sends to the staging area. When using one record at a package, we have in fact the case of not using a SFlowR. In Fig. 8 and 9, we refer to the regular operation of the source (without sending records towards the ADSA) as "plain".

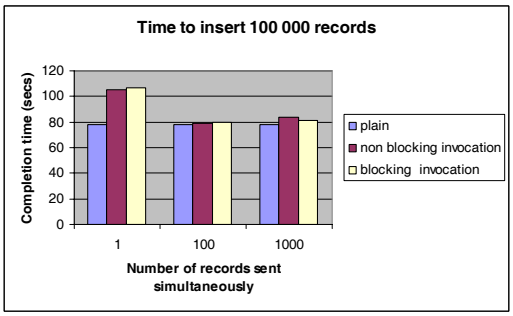


Fig. 8. Time to insert 100 000 records using two-tier topology

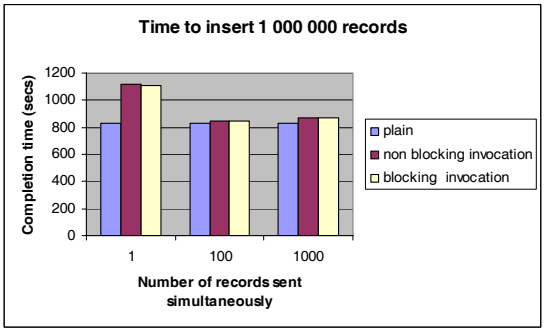


Fig. 9. Time to insert 1 000 000 records using two-tier topology

Another issue worth investigating is the isolation of the Source, ADSA and Data Warehouse layers. Therefore, we employ two modes for the operation of the staging area, to assess its impact. Each test case is examined with blocking and non-blocking invocation for the communication between the staging area and the Web Service at the data warehouse side. The staging area uses a synchronized queue. The input rate at the queue is equal to the output rate of the Legacy Application. The queue's output rate is fixed to one thousand records per second.

Figure 8 depicts the results of the experiment for 100 000 records, while Figure 9 the results for 1 000 000 records. The x-axis for Figures 8 and 9 shows the number of rows in a packet. The y-axis of the diagrams measures the throughput of inserting the records to the ISAM file.

Based on our experimental results, the following observations are made:

1. The SFlowR plays a very important role, since without it the throughput deteriorates by 34%, while using a SFlowR incurs an impact of approximately 1.7%.
2. The way that the DWFlowR is tuned does not affect the source. Regardless of using blocking or non blocking Web Service invocation at the DWFlowR, the source's throughput is the same in both cases.
3. Sending smaller packets of records performs slightly better, since in the case of 1000 records, network propagation time decreases throughput. Moreover, choosing a packet size of 100 instead of 1000 records saves buffer size at the SFlowR.
4. The cost delay ratio in terms of the size of data sent to the warehouse remains stable both in the case of 100 000 and 1 000 000 records.
5. The behavior of our system remains stable regardless of the size of data it has to handle.

4.1.4 Data Freshness

A major requirement in our setting is to achieve the maximum data freshness possible, through our framework. With a 1.7% delay at the source, the focus of interest is isolated in the side of the staging area. The goal of the next set of experiments is to measure the data freshness time provided by our application with respect to the queue emptying rate and the block retrieved from the queue. We consider as *data freshness time* the time required for a record that was inserted in the ISAM file to be transferred to the warehouse.

Specifically, we measure the overall throughput, i.e., the time needed to empty the DWFlowR's queue after the first record is sent to the warehouse. The freshness is then measured as the time needed to empty the queue, which practically stands for the response time for the last record. To perform these measurements, we assume that the legacy application sends 100 000 records to the staging area in blocks of 100 records over TCP. Also, we measure the queue length as an indicator of resource consumption at the staging area.

It is important to determine the behavior of the ADSA using data service rates close to the service rate of the source. Since our data warehouse server is faster than our source, we wanted to simulate slower performance to determine the behavior of the system in marginal conditions. Thus, we empty the queue retrieving the records from the queue using timeouts of 0.1 seconds and retrieving 100, 150 and 200 records each time and then invoking the web service, having as a source data rate approximately 1300 records per second. These are the maximum emptying rates, meaning that if the queue contains fewer records, then all the records from the queue are retrieved. We also present the results of the server operating at its top performance.

The results of emptying the queue using various rates are depicted in Figure 10. In these graphs, two other parameters play a major role. The first parameter, as indicated on the x-axis, is the time required to empty the queue. The second parameter, as shown on the y-axis, is the number of elements in the DWFlowR's queue. Figure 11 depicts the data freshness provided by our architecture.

We measure the time required to transfer all data from the staging area to the data warehouse.

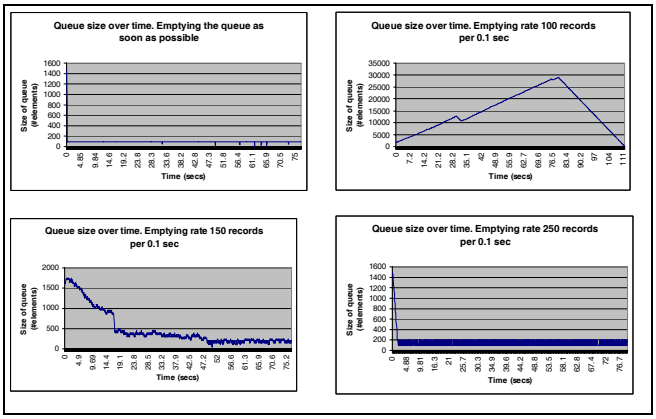


Fig 10 Queue size at the staging area emptying the queue as soon as possible

In Figure 10, the top left graph shows what happens when we let the ADSA operate fully. We can easily see that practically no queue is ever formed. The mean queue size is 100 records which is the rate of the SFlowR. In other words, the ADSA is one step later than the source, in terms of performance.

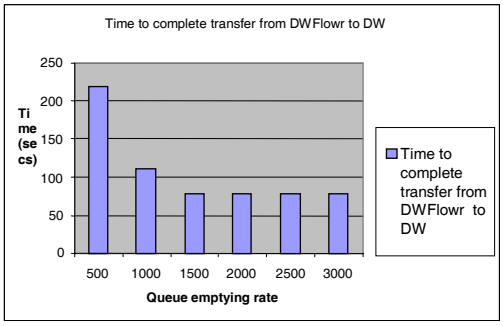


Fig. 11 Queue emptying time at the staging area.

The other three graphs show the queue sizes using service rates of 1000, 1500 and 2000 rows per second. In the first case, where the service rate is lower than the arrival rate, the queue explodes, as expected. In the second case, where we are close to the arrival rate, the queue displays a quite transient yet stable behavior. The last graph practically presents the same behavior as in the first graph even though the service rate is slightly increased compared to the case of 1500 rows per second. We have also experimented with even higher service rates i.e., up to 3000 rows per second, which still present the same behavior. We omit these results due to lack of space.

Observing the results of this set of experiments, we are led to the following conclusions:

1. We can achieve data freshness time equal to data insertion time when we *continuously* empty *a small size* queue.
2. In this case, the size of the queue is equal to the arrival rate from the source, i.e., there is practically no delay at the queue.

4.2. Operational Evaluation

In this subsection, we will use the architectural guidelines derived from the first set of experiments presented in subsection 4.1 to build an active data warehouse where we will also deploy our online ETL operations. The aim of this section is to evaluate the behavior of this fully deployed system.

4.2.1 Impact at the Source

In this paragraph, we will try to refine the results learned in 4.1. For this reason, we examine again the impact on the source system of the packet size of the SflowR. This time we will use small package sizes, as derived from the previous set of experiments.

Figure 12 shows the impact at the source using packets at the SflowR of various sizes. In general, packet sizes of over 25 records offer the least burden to the source. The smallest delay was achieved with a packet size equal to 50, where the source delay was measured to be at 5.8%.

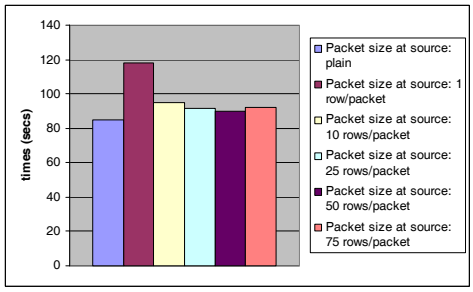


Fig. 12 Packet size of the SFlowR and impact at source

4.2.2 Data Freshness of Online ETL

In this paragraph, we deploy certain ETL scenarios and evaluate their performance compared to the theoretical analysis and in terms of data freshness. For this reason, we consider the following scenarios and their individual steps:

- Scenario (a): We simply transfer data inserted into the legacy application to the warehouse using various service rates.
- Scenario (b): (1) We filter 10% of incoming data through a selection predicate. (2) Then, we employ a surrogate key transformation to the first column of the filtered data. (3) Next, we perform a cumulative aggregation (group by with sum). (4) Finally, data are fed to the warehouse.
- Scenario (c): (1) We filter 10% of incoming data. (2) Then, we additionally filter another 2% of the remaining data. (3) Next, a surrogate key operation is applied to the first column of the data. Then, the stream is replicated along two branches.
    - For the first branch populating a materialized view, (4.1.1) a cumulative aggregation is performed and (4.1.2) data are fed to the warehouse.
    - For the second branch, populating the detailed fact table (4.2), data are fed to the warehouse.
- Scenario (d): (1) We filter 10% of incoming data. (2) We replace the values of the first field, to simulate value computations through functions. (3) A surrogate key transformation is applied. Then, the stream is replicated along two branches:
    - For the first branch, (4.1.1) a cumulative aggregation is performed first and (4.1.2) a filter (HAVING clause) rejecting 6% of the groups is applied. Then, (4.1.3) data are fed to the warehouse.
    - For the second branch, (4.2.1) a second value derivation is performed, (4.2.2) a filter rejecting 2% of detailed input data is applied and, finally, (4.2.3) data are fed to the warehouse.

In Figures 13, 14, 15 and 16 we depict the evolution of the experiments as time passes. The x-axis depicts the time points when we measured the queue length. The final time point gives the time (in seconds) required to complete the transfer from the ADSA to the Warehouse. The y-axis depicts the number of rows existing in the queue. The graphs only show the time points when our measurement showed that the queue is not empty. Each of the queues in the graph is identified by its operation name (e.g., in Figure 13, "FILTER"), possibly its selectivity (e.g., "10" for 10%) and its occurrence in the scenario (e.g., "01" for the first occurrence).

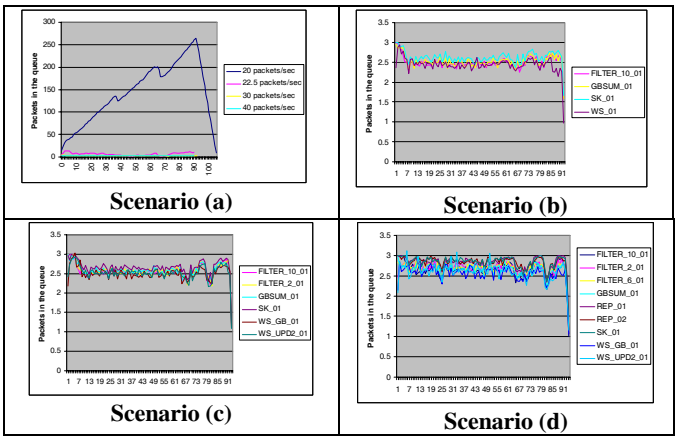


Fig. 13-16 Queues for scenarios (a), (b), (c), (d)

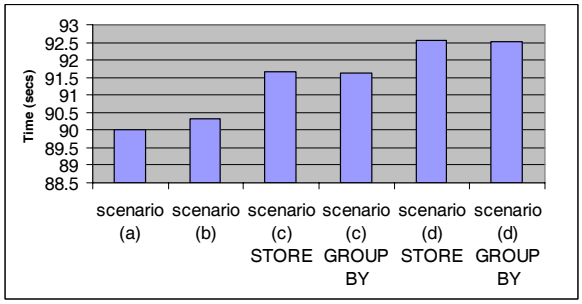


Fig. 17 Data freshness for each scenario

In all the scenarios the block size of the SFlowR was fixed at 50 rows per block. Scenario (a) was configured to use the following service rates: 20, 22.5, 30 and 40 packets per second, which represent rates of 1000, 1250, 1500 and 2000 rows per second respectively. In scenarios (b), (c) and (d) the service rates were simulated to 30 packets per second both for the ETL rates and the Web Service clients.

Finally, Figure17 summarizes the total times needed for the ADSA to transfer all data to the warehouse, for each scenario of ETL queues.

Observing the figures, we derive the following conclusions:

1. The source capability is approximately 1100 rows/sec. In scenario (a) we are led to queue explosion, when we employ service rate smaller than the source's arrival rate. Using a service rate of 1250 rows / sec, which is a setting close to the arrival rate, we can see that transient effects tend to appear, but the queue converges to steady state. By using higher service rates, 1500 and 2000 rows / sec respectively, the queue maintains its steady state.
2. In scenarios (b), (c) and (d) we observe that the entire system, as well as the queue of each operation, maintains a steady state. The number of packets in the queue is less or equal to the maximum number of packets polled simultaneously from the queue. This practically means that after each poll the queue empties and that the ADSA is only one step behind the source.
3. In Figure 17, the total time needed for the entire dataset to be transferred from the ADSA to the Warehouse is dependent on the number of the intermediate ETL operations. As the number of intermediate ETL operations that a packet has to visit increases, the total delay increases as well. Nevertheless, in our exemplary scenarios, the increase is rather small, due to the pipelining of data. The average delay per row is around 0.9 msec for all scenarios.

In Table 2 we present the comparison of our theoretical evaluation of queue length against the observed values. For lack of space, we show only the results of scenario (c) with service rate of 2000 rows/sec; all the other scenarios present identical behavior. As one can observe, in average, the theoretical prediction typically underestimates the average queue length by a very small amount (of the size of 5 records). In our detailed experiments, the system behaves in accordance with this pattern for all four scenarios, with an average error of half a packet (i.e., 25 records).

Table 2. Theoretical prediction vs. actual measurements of average queue length for scenario (c) in packets

	Measured	Theoretical Prediction	Difference
FILTER_10_01	0.160	0.056	0.104
FILTER_02_01	0.134	0.047	0.087
SK_01	0.154	0.054	0.100
GB_SUM_01	0.137	0.048	0.089
WS_GB	0.091	0.031	0.059
WS_GB_UPD	0.100	0.035	0.066

5. RELATED WORK

In this section, we present work related to our approach. Research in ETL has provided results in (a) tools [10, 21], (b) algorithms for specific tasks [7, 15, 16, 18]. Both tools and algorithms operate in a batch, off-line fashion. So far, minimum emphasis has been paid to the investigation of ETL tasks, apart from a general model for [7, 18], where ETL activities are studied under the prism of lineage or resumption of a failed process. As already mentioned, data streams [1, 5, 17] could possibly appear as the paradigm for active warehouse maintenance. So far, streams have been studied from the point of view of continuous querying, without any investigation of transformations or updates. Both our architecture and theoretical analysis could possibly be applied over streams for this purpose. To our knowledge, the only paper related to our approach is [14], where the authors apply a "white-box" (as opposed to our black box) method to determine the properties of SPJ relational operators with respect to queue theory.

Work in materialized views refreshment [12, 13, 24, 25] is orthogonal to our setting. In [13] the authors describe materialized views, their applications, and the problems and techniques for their maintenance. Novel techniques and an up-to-date survey of related work in the field are presented in [12]. Materialized views refreshment fits orthogonally with our on-line refreshment technique, since we can treat each ETL queue as a black-box process. In the context of this paper, a dedicated web service is assigned to each materialized view. Although the tuning of the system for large workloads of views is an interesting topic of research, we find this issue outside the scope of this paper.

Another area related to our approach is the one of active databases. In particular, if conventional systems (rather than legacy ones) are employed, one might argue that the usage of triggers [7] could facilitate the on-line population of the warehouse. Still, related material suggests that triggers are not quite suitable for our purpose, since they can (a) slow down the source system and (b) require changes to the database configuration [6]. In [19] it is also stated that capture mechanisms at the data layer such as triggers have either a prohibitively large performance impact on the operational system. As compared to these problems, our architecture achieves low overhead with minimal impact in the configuration of the source. We conjecture that a replication mechanism similar with the proposed one, propagating log entries towards the warehouse is a possible solution towards this problem.

6. CONCLUSIONS AND FUTURE WORK

Active Data Warehousing refers to a new trend where data warehouses are updated as frequently as possible, due to the high demands of users for fresh data. In this paper, we have proposed a framework for the implementation of active data warehousing, keeping in mind the following goals: (a) minimal changes in the software configuration of the source, (b) minimal overhead for the source due to the "active" nature of data propagation, (c) the possibility of smoothly regulating the overall configuration of the environment in a principled way. In our framework, we have implemented ETL activities over queue networks and employed queue theory for the prediction of the performance and the tuning of the operation of the overall refreshment process. In terms of data freshness, source overhead and minimal impact of software configuration the results seem satisfactory. A summary of the lessons learned is as follows:

- In terms of architecture, isolating the ETL tasks in a special-purpose area, either in the warehouse, or in an intermediate tier, guarantees both minimum performance overhead at the source and the possibility of regulating the flow towards the warehouse target tables.
- Queue theory can be successfully employed as the theoretical background for the estimation of the response of the active staging area. The system reaches a steady state quite close to the predicted behavior. Freshness is quite satisfactory too.

- The overall overhead at the source side is around 1.7% and the amount of code modification is around 100 lines, without affecting applications.
- Tuning the network-related parameters helps. TCP should be used instead of UDP, due to the packet loss of the latter. Organization of rows in blocks, both at the source and the ADSA side increases performance.

Future work includes several directions. A first line of research would have to do with the failure management of the components of the environment, to determine safeguarding techniques and fast resumption algorithms for the event of a failure. Further tuning can be made, by testing multiple concurrent loading sources for the warehouse. Also, the case of materialized aggregate views and schema evolution poses interesting challenges in this context.

7. ACKNOWLEDGMENTS

E. Papapetrou has helped with comments on issues of queue theory and implementation. This research has been partially supported from the European Commission and the Greek Ministry of Education through the Pythagoras Program.

8. REFERENCES

[1] Daniel J. Abadi, Don Carney, Ugur Çetintemel, et al. Aurora: a new model and architecture for data stream management. *The VLDB Journal, 12(2)*, 120-139, 2003.

[2] G. Alonso, F. Casati, H. Kuno, V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer-Verlag, 2003.

[3] J. Adzic, V. Fiore. Data Warehouse Population Platform. In *Proc. 5th Intl. Workshop on the Design and Management of Data Warehouses (DMDW'03)*, Berlin, Germany, 2003.

[4] Apache Software Foundation. Axis. Available at http://ws.apache.org/axis/

[5] S. Babu, J. Widom. Continuous Queries over Data Streams. *SIGMOD Record 30(3)*, 109-120, 2001.

[6] Donald Burleson. New Developments In Oracle Data Warehousing. Available at: http://dba-oracle.com/oracle_news/2004_4_22_burleson.htm

[7] Stefano Ceri, Jennifer Widom. Deriving Production Rules for Incremental View Maintenance. In *Proc. VLDB, Barcelona Spain, September 1991*, 577-589

[7] Yingwei Cui, Jennifer Widom. Lineage tracing for general data warehouse transformations. *The VLDB Journal 12(1)*, 41-58, 2003.

[9] W. Duquaine Web Services Ruminations. Presentation at *High Performance Transaction Systems Workshop (HPTS'03)*. Asilomar Conference Center, California, October 12-15, 2003. Available at http://research.sun.com/hpts2003/

[10] Galhardas, H., Florescu, D., Shasha, D., and Simon, E.. Ajax: An Extensible Data Cleaning Tool. *In Proc. ACM SIGMOD*, Dallas, Texas, May 2000, p. 590.

[11] D. Gross, C. Harris. *Fundamentals of Queuing Theory*. Wiley, 3rd Edition, 1998.

[12] H. Gupta and I.S. Mumick. Incremental Maintenance of Aggregate and Outerjoin Expressions. To appear in Information Systems, 2004.

[13] Ashish Gupta, Inderpal Singh Mumick. Maintenance of Materialized Views: Problems, Techniques, and Applications. *Data Engineering Bulletin 18(2)*, 3-18, 1995.

[14] Qingchun Jiang, Sharma Chakravarthy. Queueing analysis of relational operators for continuous data streams. In *Proc. CIKM*, New Orleans, Louisiana, USA, November 2003, 271-278.

[15] Wilburt Labio, Jun Yang, Yingwei Cui, Hector Garcia-Molina, Jennifer Widom: Performance Issues in Incremental Warehouse Maintenance. In *Proc. VLDB*, Cairo, Egypt, September 2000, 461-472.

[16] Wilburt Labio, Hector Garcia-Molina: Efficient Snapshot Differential Algorithms for Data Warehousing. In *Proc. VLDB*, Mumbai, India, September 1996, 63-74.

[17] D. Lomet, J. Gehrke. Special Issue on Data Stream Processing. *Data Engineering Bulletin*, *26(1)*, 2003.

[18] Wilburt Labio, Janet L. Wiener, Hector Garcia-Molina, Vlad Gorelik. Efficient Resumption of Interrupted Warehouse Loads. In *Proc. of ACM SIGMOD*, Dallas, Texas, USA, May 2000, 46-57.

[19] On-Time Data Warehousing with Oracle10g - Information at the Speed of your Business. An Oracle White Paper. August 2003. Available at http://www.oracle.com/technology/products/bi/pdf/10gr1_twp_bi_ontime_etl.pdf

[20] P. Graf. The Program Base Library. Publicly available through http://mission.base.com/peter/source/

[21] Vijayshankar Raman, Joseph M. Hellerstein: Potter's Wheel. An Interactive Data Cleaning System. In *Proc. VLDB*, Rome, Italy, September 2001, 381-390.

[22] C. White. Intelligent Business Strategies: Real-Time Data Warehousing Heats Up. *DM Peview*, August 2002. Available at http://www.dmreview.com/article_sub.cfm?articleId=5570

[23] A. Willig. Performance Evaluation Techniques. Available at http://www-ks.hpi.uni-potsdam.de/docs/engl/teaching/pet/ss2004/script.pdf, 2004.

[24] Yue Zhuge, Hector Garcia-Molina, Joachim Hammer, Jennifer Widom: View Maintenance in a Warehousing Environment. In *Proc. of ACM SIGMOD*, 1995, 316-327.

[25] Xin Zhang, Elke A. Rundensteiner: Integrating the maintenance and synchronization of data warehouses using a cooperative framework. *Information Systems 27(4)*, 219-243, 2002.

An Event Based Framework for Improving Information Quality That Integrates Baseline Models, Causal Models and Formal Reference Models *

Joseph Bugajski
Visa International
PO Box 8999
San Francisco, CA 94128
jmbugajski@yahoo.com

Robert L. Grossman †
and Eric Sumner
Open Data Partners
1145 Westgate Street
Oak Park, IL 60301
{rlg1@, esumner@}opendatagroup.com

Zhao Tang
Bearing Point
1676 International Drive
McLean, VA 22102
tao.zhang@bearingpoint.com

ABSTRACT

We introduce a framework for improving information quality in complex distributed systems that integrates: 1) Analytic models that describe baseline values for attributes and combinations of attributes and components that detect statistically significant changes from baselines. These models determine whether a significant change has occurred, and if so, when. 2) Casual models that help determine why a statistically significant change has occurred and what its impact is. These models focus on the reasons for a change. 3) Formal business and technical reference models so that data and information quality problems are less likely to occur in the future. In this note, we focus on the first two types of models and describe how this framework applies to data quality problems associated with electronic payments transactions and highway traffic patterns.

1. INTRODUCTION

In this note, we introduce a framework for monitoring, exploring and ameliorating the information quality of event based data. We are interested in data and information quality problems for complex, distributed real time systems. Here are two motivating examples that are described in more detail below.

The first example is the processing of electronic payments. A payments card transaction is an example of an event and involves several parties, namely the cardholder, the merchant, the merchant's bank, the cardholder's bank and the payment processor. Each of these independent parties is involved in the decision of whether to accept the transaction, decline the transaction, or request further information about the transaction. Poor data and information quality can increase the rate of improper declines and improper approvals.

The second example is the real time analysis of traffic patterns over a metropolitan region in order to quickly identify accidents and other anomalous behavior. In this example, we assume that traffic sensors produce real time information about the speed and volume of traffic. The resulting sensor readings are examples of events which are aggregated to produce features summarizing the traffic at a particular time and location. Traffic patterns can be quite complex and vary with the time, location, weather, local events, etc. Obtaining accurate data can be very challenging.

There are several challenges common to both of these examples:

1. The data sizes are large and the data is heterogeneous.
2. The data is produced and processed by several different parties and this sometimes introduces data and information quality issues.
3. The data and system is sufficiently complex that establishing baseline data quality and information levels can be quite challenging.

In this note, we describe a framework for monitoring, exploring and ameliorating the data and information quality for systems with these types of challenges. The framework has four components:

Building Baselines. The first component of the framework is an analysis engine that:

1. analyzes event based data
2. divides the event based data into appropriate segments
3. computes features or states from these events for each segment
4. and from these features estimates appropriate baselines for each segment.

The idea is that although the system has a whole may be quite complex by dividing the data into enough cells (by

*This work was supported in part by the Visa International Data Interoperability Program and the U.S. Army Pantheon Project.

†Robert L. Grossman is the corresponding author. He is also a faculty member at the University of Illinois at Chicago.

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

restricting to appropriate ranges of values along each dimension), the data becomes homogeneous enough to analyze, addressing the first challenge. In this context, we call each such cell a *segment* (as in segmented modeling). In our experience, analyzing appropriate features associated with various entities of interest instead of directly analyzing the events themselves helps address the second challenge.

Monitoring. The second component of the framework is a monitor that:

1. monitors streams of event based data
2. computes summary or state information,
3. uses this information as input to statistical measures and models
4. compares the outputs of the measures and models to previously computed baselines, and
5. issues alerts in case of statistically significant deviations.

The goal of monitoring is to determine *whether* a statistically significant change has occurred. In other words, rather than starting with a certain expectation of data or information quality, the approach is to detect as quickly as possible changes in data or information quality, addressing the third challenge.

Root Cause Analysis. The third component of the framework is a process for exploring the monitored data to understand casual relationships between data and defined outcome variables. A variety of techniques can be used to understand causality, including contingency tables [1], discriminant analysis, regression, and classification and regression trees [13]. The challenge is to understand whether different variables are causally related or simply correlated.

Here is a simple example from the analysis of payments card transactions: The decline rate of transactions is an outcome variable that has obvious business significance. Some declines are due to insufficient funds or fraudulent usage, while others are due to data quality problems. Errors in how a merchant processor sets up an e-commerce system can lead to hidden data quality problems and higher than usual declines. The role of the root cause analysis process is to understand some of the casual reasons for statistically significant changes in baselines. In other words, the goal of root cause analysis is to determine *why* something has happened.

Amelioration. Once one one or more root causes are identified, the goal of the fourth component of the framework is to take actions to ameliorate the problems. In the example, above this may involve educating the merchant processor so that the identified data quality problems do not occur in the future.

In our experience, data quality problems for complex distributed systems are often the result of documentation that is hard to understand or difficult to interpret. We have been exploring the use of model driven architecture [7] to provide formal business and technical reference models and methods that can directly address this difficulty.

In this paper, we describe this framework and provide some high level experiences of some of the implementations we have done.

Although monitoring, causal analysis and amelioration are components for several different data and information quality methodologies [6], [14], [15], as best as we can tell from reading the literature, our paper makes the following contributions:

1. Most data and information quality methodologies [14], [17], [15] do not distinguish carefully between transaction or event data and summary or profile data that is aggregated from it. This is an important distinction for our targeted applications. As a simple example, the data and information quality issues are quite different for payments card transactions and summary information at the merchant, account, issuer, or acquirer level.
2. A common approach to data and information quality is to measure the quality of data along several dimensions. For example, accuracy, completeness, validity, timeliness, etc. (see, for example, [20]). In contrast, our focus is not on the dimensions themselves but on effective procedures for creating small cells or segments of data (defined by dimensional ranges) that have both business and statistical significance and building effective baselines for each cell. For example, we view data for a transaction process as being naturally divided into cells by logical entity (issuer, acquirer, type of payments card or payment product) and temporal entity (weekday, holiday, weekend, etc.)
3. Our methodology is closely tied to standards, in particular, the Predictive Model and Markup Language or PMML, that dominant standard for statistical and data mining models. This has several important implications. In particular, this allows us to instantiate a data quality in a standards based fashion as an XML file.

2. RELATED WORK

There is now quite a bit of research in the field of data and information quality, and several books [18] and [3]. In this section, we briefly discuss some of the research that is most directly relevant.

Our approach to data and information quality is statistical. This tradition goes back at least to Deming [4]. In particular, our focus is on establishing baselines and measuring statistically significant deviations from baselines. This is a standard approach in change detection [2]. In contrast, many approaches for data and information quality are business systems or engineering based (see for example, [14] or [18]).

Once deviations from baselines are detected, a statistical analysis is undertaken to try to determine the underlying reasons. Today, there are a wide variety of approaches for trying to determine causality, including root cause analysis [19], contingency tables [1], discriminant analysis, regression, and classification and regression trees [13].

A common approach to data and information quality is to measure the quality of data along several dimensions. For example, DOD Guidelines recommend using accuracy, completeness, consistency, timeliness, uniqueness and validity. As another example, Strong et. al. [20] introduce 16 dimensions organized into four categories (intrinsic information quality, contextual information quality, representational informational quality, and accessibility information quality).

In this note, we use some, but not all, of these standard dimensions. In particular, most of the work described below are based on metrics measuring completeness, consistency, and validity.

In this note we distinguish formally between input events and persistent states. Although this is standard in dynamical systems, automata theory, and control theory, but does not appear to be a standard approach in statistics or data mining [11].

Most data and information quality methodologies [14], [17], [15], [6] include components for defining, measuring, analyzing, and improving data and information quality issues, as our does. On the other hand, the approach sketched below differs in two significant ways from [14], [17], [15], [6] and related work:

1. Our approach is closely tied to standards based architectures. As many approaches do today, we employ a data warehouse. In addition, we employ a monitor for monitoring streaming data, a component for building baselines, and a scoring engine [12] for measuring the deviation of the streaming data from the baseline.

2. Second, our approach is closely tied to standards for data mining and statistical models [10] and [5], such as the XML-based Predictive Model Markup Language.

3. EXAMPLES

We have applied the framework described here to several examples, including those involving payments card transactions, highway traffic data, and multi-modal sensor data. In this section, we provide a bit of background for two of these examples in order to make this note more self contained.

Here is a simplified description of some of the steps involved in a payments card transaction.

1. A cardholder (the card has an account number) purchases an item at a merchant.

2. The merchant has a relationship with a bank called the acquiring bank, which agrees to process the payments card transactions for the merchant. The acquiring bank provides the merchant with a terminal or other system to accept the transaction and to process it.

3. The acquiring bank has a relation with financial payment system, such as those operated by Visa, MasterCard, Discover, etc. The transaction is processed by the acquiring bank and passed to the payment system.

4. The payment system the transaction and passes the transaction to the bank that issued the payments card to the card holder (the issuing bank). In other words, one of the essential roles of the payment system is to act as an intermediary between the acquirer and the issuer.

5. The issuing bank processes the transactions and determines if there are sufficient funds for the purchase, if the card is valid, etc. If so, the transaction is authorized; the transaction can also be declined, or a message returned asking for additional information. In each of these cases, the path is reversed and the transaction is passed from the issuing bank back to the payment system, from the payment system back to the merchant bank, and from the merchant bank, back to the merchant.

One of the challenges of a problem like this is to monitor in real time data and information quality problems for the various different parties when the data is processed transaction by transaction. Our approach is to use event based processing model and to create different summary or feature vector for each entity of interest. In this case, this includes the cardholder, the merchant, the acquirer, the issuer, and the payment system.

As another example, consider the problem of understanding highway traffic congestion. The Gateway System employs over 800 sensors to collect volume, speed, and occupancy data in a three state, fifteen county Gary-Chicago-Milwaukee (GCM) corridor [16]. The Pantheon Gateway Testbed augments this data with data about weather, special events that may effect highway conditions, and related information.

Here is a question that can be posed using this data: On a Monday, around 7 am, that is not a holiday, and when it is beginning to rain lightly, what is the average speed and volume for traffic on Interstate - 290 near the Austin exit? What will the average speed and volume be around 8 am if the rain continues?

This is an important motivating question and suggests our approach. Rather than try to understand data and information quality problems for the system as a whole, our approach is divide the data into relatively homogeneous segments or cells (such as traffic on Mondays around 7 am with light rain) and to establish appropriate baselines for each such segment. With this knowledge, understanding data and information quality problems becomes much easier.

4. OVERVIEW

Broadly speaking, our technical approach is as follows:

1. We assume that data consists of events, such as transactions or sensor readings. Events are first divided into segments or cells that are relatively homogeneous. An event can be associated with multiple segments. Of course, there are many different ways of doing this and a discussion of these is outside the scope of this paper.

2. For each segment, events are processed to update state or feature vectors containing persistent features associated with the events. This is described in more detail in the section below.

3. For each segment, appropriate baselines are established for collections of state or feature vectors. Baselines may be temporal, geospatial, logical, or some combination. Determining good break points for dimensions appears to be a difficult problem and is outside the scope of this paper.

4. Deviations from baselines are detected using simple threshold models or more complex change detection models [2]. Deviations are used to determine as quickly as possible whether something has changed.

5. Separately, casual analysis is used to determine whether conditions or combinations of conditions are likely to

effect outcome or impact variables. To begin n by m contingency tables are used as a starting point for this analysis [1]. This is supplemented as required by discriminant analysis, linear regression, and nonlinear regression techniques, such as classification and regression trees [13].

6. When important casual conditions are identified, formal models [7] are used to begin to improve the condition. The use of formal models for this purpose is also outside the scope of this paper.

5. EVENT BASED DATA PROCESSING

It is useful when developing baselines to distinguish between data and derived attributes following [5].

A *data attribute* is simply an attribute present in the data itself, while a *derived attribute* is an attribute derived from the data or aggregations of the data. For example, given a payments card transaction the raw amount of the transaction is in the data itself, while currency related attributes, interchange fees, the amount of transactions for an account holder during the past hour, the number of declined transactions that are e-commerce-related, etc. are all examples of derived attributes.

In this note, we follow an event based approach to analyze information [11]. We assume that we are given:

- A stream of events α_1, α_2, ... in $\mathbf{R}^m$. Attributes in the events are data attributes.
- A finite collection ξ of feature vectors (also called state vectors)

$$\xi = \{x_1, x_2, \ldots, x_n \in \mathbf{R}^N\}.$$

- An update rule (denoted dot) specifying how an event α updates the collection of feature vectors

$$\xi' = \alpha \cdot \xi.$$

  Attributes in the feature or state vectors consist of derived attributes formed from the event data through transformations and aggregations.

- A function of the state space

$$f : \mathbf{R}^N \longrightarrow \mathbf{R}^1$$

  representing a statistical or data mining model producing scores or other outputs.

We illustrate this using our running example of payments card transactions. In this case, the events are payments card transactions, while the state vector represents information associated with a related entity, such as a payments card or issuing bank. For example, if the state vector represents a payments card transaction, the a component of the state vector might be the number of transactions during the the previous 60 minutes. If the state vector represents an issuing bank, then a component of the state vector might be the number of declined transactions during a day. In both cases, the model might be a change detection model indicating that the observed feature is statistically different than a previously computed baseline level.

For another example, assume that events consist of sensor readings for a collection of sensors and that there is a feature vector for each sensor that maintains the number of readings, the average of the readings, the min sensor reading, and the maximum sensor reading for each sixty minute period, for each of the $168 = 7 \times 24$ sixty minute periods during a seven day week. Here the update rule, updates the features corresponding to the appropriate sixty minute period.

6. BASELINE MODELS

In this section, we review a standard approach for detecting deviations from baselines [2]. In the methodology described here, this is applied to each segment or cell separately. We assume that one have mean and variances representing normal behavior and behavior that is not normal.

More explicitly, assume we have two Gaussian distributions with mean μ_i and variance σ_i^2, $i = 0, 1$.

$$f_i(x) = \frac{1}{\sqrt{2\pi}\mu_i} \exp \frac{-(x-\mu_i)^2}{2\sigma_i}$$

The log odds ratio is then given by

$$g(x) = \log \frac{f_1(x)}{f_0(x)}.$$

and can now define a CUSUM algorithm as follows [2]:

$$Z_0 = 0.$$

$$Z_n = \max\{0, Z_{n-1} + g(x_n)\}.$$

Streaming data is compared to existing baseline data and deviations are noted and flagged for investigations. Baseline models like these are used to determine whether something has changed, and, if so, when it changed.

Baseline models aggregate data along several dimensions (in the running example of payments card transactions, baseline models aggregate data by issuer, acquirer, region, temporal period, type of transaction, etc.) At too high a level of granularity, too much information is lost. At too fine a level of granularity, it is too difficult to discern what is important.

For example, it is important to know that the overall authorization rate is 93%, but this doesn't easily lead to actions that improve the authorization rate. On the other hand, knowing that the authorization rate for transactions with inconsistent point of sales data is 20% higher than the average provides some very important information. The challenge with baseline models is to choose the right of level of granularity so that the baselines are meaningful and can uncover opportunities for improvement. From this perspective, advanced baseline models can dive down into the data to uncover homogeneous pockets of data and establish appropriate baselines for each pocket.

The Predictive Model Markup Language or PMML is an XML based language to describe statistical and data mining models. As part of the work described here we have produced PMML models for baselines and to detect deviations from baselines. Using the terminology of [5], an application that produces baselines is a PMML producer and an application that monitors baselines is a PMML consumer.

7. ROOT CAUSE ANALYSIS

Different applications structure the root cause analysis differently. For example, when analyzing payment data the current approach consists of two steps:

1. In the first step, the relation between different conditions and different outcome or impact variables is examined using to generate alerts, together with confidence levels. The goal of this step is to provide an initial identification of conditions that are correlated with variables of business interest.

2. In the second step, an investigation is undertaken involving subject matter experts to explore the relationship and to determine whether the relation is *casual*, and, if so, to estimate its business impact.

We now briefly describe each of these steps in the running example of payments card transactions.

A simple way to analyze data is to use data and derived attributes to define conditions and then to examine the relation between the conditions and certain outcomes using contingency tables [1]. Recall that contingency tables capture the relation of two categorical variables. See the Table below for a simple example. In addition to contingency tables, we currently beginning to explore multivariate techniques, such as discriminant analysis or classification trees [13] to generate alerts.

Alert conditions are defined by specifying values or ranges of values for data or derived fields. Defining binary indicator variables is a very simple way of defining conditions. Here is a simple example. A transaction has a field indicating that it is e-commerce related. For example, an indicator attributed can be defined by defining a condition to be 1 if the transaction is e-commerce related in this sense and 0 otherwise. As another example, a payments card transaction also has a field indicating the type of merchant. An indicator variable can be defined if the type of merchant is a casino and 0 otherwise. More complex types of conditions can also be defined. For example, conditions with three, four or more different values can also be defined. Conditions defined in these ways are examples of statistical factors.

Outcome and impact attributes can be data attributes, but are generally derived attributes. Examples include a binary variable indicated whether a financial transaction is approved or not. As another example, a binary indicator variable indicating whether a transaction is cleared, or whether or not a transaction is associated with a charge back or not.

	Outcome - State 1	Outcome - State 2
Alert Condition Present	n_{11}	n_{12}
Alert Condition Not Present	n_{21}	n_{22}

Table 1: A 2x2 contingency table that is sometimes a helpful step in the root cause analysis of alerts.

In addition to baseline models, our framework also uses more complex statistical to help determine what conditions and combination of conditions are likely to result in certain outcomes, such as a decrease in authorizations. In conjunction with this, our framework also employs an investigative process involving subject matter experts to help determine why an outcome variable (such as the approval rate or charge back rate) has changed and if so what the impact is? We call these *casual models*.

8. STATUS

To date, we have undertaken several projects using this methodology. Here we give a brief summary of the status of two of these.

Payments card transactions. We have begun to analyze data and information quality problems associated with declines for a large financial transaction processor. The status is as follows: some measures for incomplete, invalid, and inconsistent fields have been developed. Using this measures, we are currently developing baselines and identifying combinations of conditions capturing common data and information quality problems. We are also examining casual relations between these conditions and impact variables, such as the rate of declines. Finally, preliminary business and technical reference models for some of the more important fields have been developed [7].

Highway traffic data. The Gateway System collects near real time data from over 800 highway traffic sensors covering the three state, fifteen county Gary-Chicago-Milwaukee (GCM) corridor. This data is archived by the Pantheon Gateway Project [16] and overlaid with data about special events, such as concerts or sports events, and data about the weather. In addition, data about accidents is collected. Using this data, we have established preliminary baselines used a real time scoring engine employing PMML-based change detection models to detect statistically significant changes from these baselines. To date, CUSUM-based and threshold based change detection models [2] have been developed and deployed. Currently, casual models using tree-based classifiers are being developed to try determine semi-automatically whether deviations from baselines are due to chance, unusual weather, special events, or accidents.

Publicly available data and information about the first projects is rather limited due to its confidential nature. On the other hand, data for the third project is publicly available from the web site [16].

9. CONCLUSION

In this note, we have introduced a framework consisting of four steps that can help identify and ameliorate data and information quality problems for complex, distributed systems.

Our assumption is that the data is event based and heterogeneous. In a preliminary step, we divide the data into more homogeneous cells or segments and aggregate the data into feature or summary vectors attached to entities of interest.

1. The first component statistically analyzes each segment and produces a baseline.

2. The second component monitors the event stream in real time and compares computed quantities of each interest in each segment to historical baselines. Deviations result in request for an investigation (an alert). This component detects whether something has happened.

3. The third component is a root cause analysis which seeks to identify the root cause of each alert. This involves subject matter experts. This component determines why something has happened and, if so, what its impact is.

4. The fourth component employs formal models [7] to reduce the likelihood that similar problems will happen in the future.

This framework has been applied in several different domains. In this paper, we discussed two of these: understanding data and information quality problems for payments card transactions and for highway traffic data.

10. REFERENCES

[1] Alan Agresti, An Introduction to Categorical Data Analysis, John Wiley and Sons, Inc., New York, 1996.

[2] M. Basseville and I. V. Nikiforov. Detection of Abrupt Changes: Theory and Application. Prentice Hall, 1993

[3] Tamraparni Dasu and Theodore Johnson, Exploratory Data Mining and Data Cleaning, Wiley, 2003.

[4] W. Edwards Deming, Elementary Principles of the Statistical Control of Quality: A Series of Lectures, JUSE, Tokyo, 1952.

[5] Data Mining Group, The Predictive Model Markup Language, Version 3.0, retrieved from www.dmg.org on March 20, 2005.

[6] DOD Guidelines on Data Quality Management (Summary), retrieved from tricare.osd.mil/rm/documents/fa/ DoDGuidelinesOnDataQualityManagement.pdf on March 20, 2004.

[7] David S. Frankel, Model Driven Architecture, Wiley Publishing Inc., Indianapolis, 2003.

[8] Glenn W. Goodman Jr., Taming the River of Data: New Software Tools Fuse Intelligence From Many Sources, Defense News, March 14, 2005.

[9] Robert L. Grossman, H. Bodek, D. Northcutt, and H. V. Poor, Data Mining and Tree-based Optimization, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, E. Simoudis, J. Han and U. Fayyad, editors, AAAI Press, Menlo Park, California, 1996, pp 323-326.

[10] Robert Grossman, Mark Hornick, and Gregor Meyer, Data Mining Standards Initiatives, Communications of the ACM, Volume 45, Number 8, 2002, pages 59-61

[11] Robert L. Grossman and R. G. Larson, An Algebraic Approach to Data Mining: Some Examples, Proceedings of the 2002 IEEE International Conference on Data Mining, IEEE Computer Society, Los Alamitos, California, 2002, pages 613-616.

[12] Robert L. Grossman, Alert Management Systems: A Quick Introduction, in Managing Cyber Threats: Issues, Approaches and Challenges, edited by Vipin Kumar, Jaideep Srivastava, Aleksandar Lazarevic, Kluwer Academic Publisher, 2004.

[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, The Elements of Statistical Learning, Springer, New York, 2001.

[14] Yang W. Lee, Diane M. Strong, Beverly K. Kahn, Richard Y. Wang, AIMQ: A Methodology for Information Quality Assessment, Information and Management, December 2002, Volume 40, Issue 2, pages 133–146.

[15] Ken Orr, Data Quality and Systems, Communications of the ACM, Volume 41, Number 2, 1998, pages 66–71.

[16] Pantheon Gateway Testbed, retrieved from highway.ncdm.uic.edu on March 20, 2005. (A SVG plug in for your browser is required to see the map.)

[17] Leo L. Pipino, Yang W. Lee and Richard Y. Wang, Data Quality Assessment, Communications of the ACM, Volume 45, Number 4, 2002, pages 211–218.

[18] Thomas C. Redman, Data Quality: The Field Guide, Digital Press, Boston, 2001.

[19] James J. Rooney and Lee N. Vanden Heuvel, Root Cause Analysis for Beginners, Quality Prgoress, 2004, pages 45–53.

[20] D. M. Strong, Y.W. Lee and R.Y. Wang, Data Quality in Context, Communications of the ACM, Volume 40, Number 5, 1997, pages 1030-110.

[21] Shawn Turner, Defining and Measuring Traffic Data Quality, Proceedings of the Traffic Data Quality Workshop, Washington, DC, December 31, 2002.

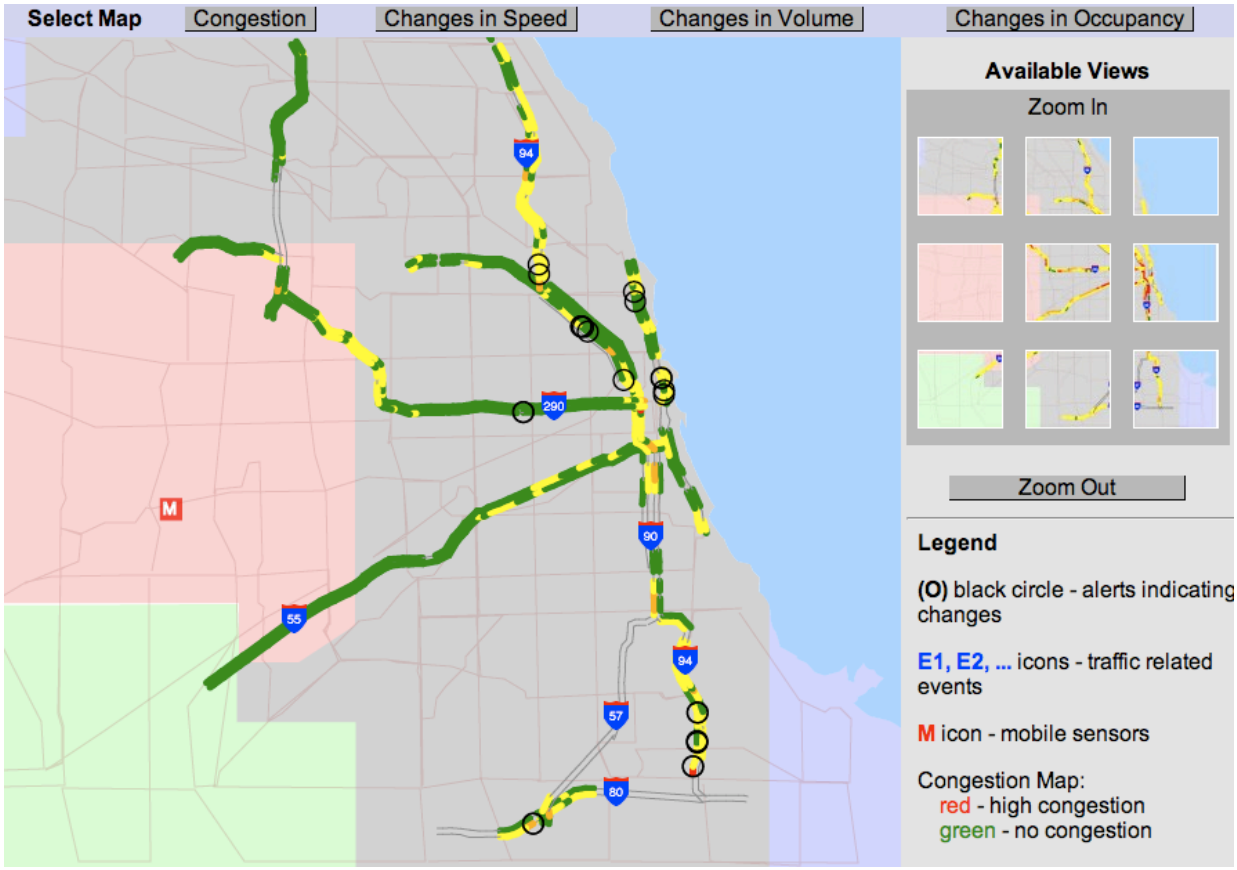


Figure 1: We have applied the framework described here to detect real time deviations from baselines from multi-modal highway data collected from over 800 highway traffic sensors in the greater Chicago region.

Exploiting relationships for object consolidation*

Zhaoqi Chen Dmitri V. Kalashnikov Sharad Mehrotra

Computer Science Department
University of California, Irvine

ABSTRACT

Data mining practitioners frequently have to spend significant portion of their project time on data preprocessing before they can apply their algorithms on real-world datasets. Such a preprocessing is required because many real-world datasets are not perfect, but rather they contain missing, erroneous, duplicate data and other data cleaning problems. It is a well established fact that, in general, if such problems with data are not corrected, applying data mining algorithm can lead to wrong results. The latter is known as the "garbage in, garbage out" principle. Given the significance of the problem, numerous data cleaning techniques have been designed in the past to address the aforementioned problems with data.

In this paper, we address one of the data cleaning challenges, called *object consolidation*. This important challenge arises because objects in datasets are frequently represented via descriptions (a set of instantiated attributes), which alone might not always uniquely identify the object. The goal of *object consolidation* is to correctly consolidate (i.e., to group/determine) all the representations of the same object, for each object in the dataset. In contrast to traditional domain-independent data cleaning techniques, our approach analyzes not only object features, but also additional semantic information: *inter-objects relationships*, for the purpose of object consolidation. The approach views datasets as attributed relational graphs (ARGs) of object representations (nodes), connected via relationships (edges). The approach then applies graph partitioning techniques to accurately cluster object representations. Our empirical study over real datasets shows that analyzing relationships significantly improves the quality of the result.

1. INTRODUCTION

Nowadays data mining techniques are widely used to analyze data for scientific applications and business decision making. To build proper models and compute accurate results it is important that analyzed datasets are accurately represented and interpreted. Many real-world datasets however are not perfect, they frequently contain various data cleaning issues such as incomplete, erroneous and duplicate data, which need to be addressed before data mining techniques can be applied. As a result, data mining practitioners frequently spend significant effort on preprocessing of data to address cleaning issues that exist in their datasets, to ensure high quality of the results.

In this paper, we address one common data cleaning challenge known as *object consolidation* [8, 20, 22, 25]. It arises most frequently when the dataset being processed is constructed by merging various data sources into a single unified database, such as by crawling the web. In many real-world datasets objects/entities are not represented by unique identifiers, instead an object is represented by a *description* (a set of instantiated attributes), used in a certain *context*, which may lead to ambiguity. An object might have multiple different representations in the dataset and also an object representation, in general, might match the description of multiple objects instead of one. The goal of object consolidation is to correctly group all the representations that refer to the same object.

For example, consider a database that contains information about two people: 'John A. Smith' and 'John B. Smith'. Firstly, entity 'John A. Smith' might have multiple representations throughout the dataset: e.g., 'John Smith', 'J. Smith' 'John Smithx' (a misspelled representation). Secondly, representation 'J. Smith' can refer to both 'John A.' and 'John B.' Smith, so one representation matches the descriptions of multiple entities. Finally, the fact that there are only two 'John Smith"s in the dataset might not be known in general. Thus, for this example the goal is to determine that all 'John Smith' representations should be clustered into two groups and then assign them to groups such that all representations for 'John A. Smith' are in one group and for 'John B. Smith' in the other. Sometimes it is possible to infer more attributes/information from the *context* in which representations appear. For example, for 'J. Smith' used in a specific context, it might be known that the mentioned 'J. Smith' works at MIT. This context information can be potentially used to consolidate representations better.

Let us use an example to demonstrate the implication of applying data analysis techniques on datasets where the object consolidation problem is not resolved correctly. Consider the task of computing author impact in a citation network using a simple citation-count statistic. That is, the

*RelDC project (`http://www.ics.uci.edu/~dvk/RelDC`)

†This work was supported by NSF grants 0331707, 0331690

IQIS 2005, June 17, 2005, Baltimore, MD, USA

task might be to compute the impact of 'John A. Smith' by counting the number of citations of his publications. This simple task might be more difficult than it seems due to the problem with representations identified above. Notice, even though the representations appear in some context, the information about the object available from the context might be of a limited nature, which makes the object consolidation task challenging. For instance, the only direct information available about the authors, for some of the publications, might be only their first initials and last names. Because of such problems with representations, some of the papers written by 'John A. Smith' might be wrongfully assigned to other authors and some of the papers written by other authors might be assigned to 'John A. Smith'. Thus, the impact of 'John A. Smith', computed on such a dataset, can be very different from the real one.

While the object consolidation problem exists in different domains, in this paper we will often use citation networks, like in the example above, to illustrate our domain-independent approach.

The problem of object consolidation is related to the problem of *record deduplication* or *record linkage* [1, 10, 13, 19, 23] that often arises when multiple tables (from different data sources) are merged to create a single database. The causes of record linkage are similar, i.e. differences in representation of objects across different datasets, entry errors, etc. The difference between the two problems is that while record linkage deals with records in a table, object consolidation deals with entities/objects – a semantic concept of a higher level. In record linkage it is often assumed that many attributes are available in each record, which are very effectively employed for deduplications. In object consolidation, however, very few attributes can be available, thus making the problem more challenging.

Another related problem is the problem of *reference disambiguation* [14, 18]. In the problem of reference disambiguation the goal is to match object representations with the list of possible objects which is known in advance and known to be clean. The requirement of having such a clean list of objects limits the applicability of reference disambiguation. As a rule, each instance of the reference disambiguation problem can be formulated as an instance of the object consolidation problem, while the reverse is not true. That is, the object consolidation problem is more general.

Most of the traditional domain-independent data cleaning techniques belong to the class of *feature-based similarity (FBS)* methods.[1] To determine if two objects/records are the same they employ a similarity function that compares values of object/record attributes (features) for the purpose of deduplication. The values of the attributes of an object are typically derived from the object representation and the context in which it is used. In this paper, we study a domain-independent approach that utilizes not only features but also additional semantic information present in datasets: inter-object (chains of) relationships. For instance, 'J. Smith' might be used to refer to an author in the context of a particular publication. This publication might also have more authors, which can be linked to their affiliated organizations and so on, forming a web of entities inter-connected via relationships. The knowledge of relationships can be exploited alongside attribute-based similarity resulting in improved accuracy of object consolidation. Our approach is based on the following hypothesis, which is referred to as the Context Attraction Principle (CAP):

The CAP hypothesis:

- *if two representations refer to the same entity, there is a high likelihood that they are strongly connected to each other through multiple relationships, implicit in the database;*
- *if two representations refer to different entities, the connection between them via relationships is weak, compared with that of the representations that refer to the same entity.* □

Our approach views the underlying database as an attributed relational graph (ARG), where nodes correspond to object representations and edges correspond to relationships. Our technique first uses feature-based similarity, to determine if two representations can refer to the same objects. If, based on the FBS similarity, two representations can refer to one object, then the relationships between those representations are analyzed to measure the *connection strength* between them. Graph partitioning techniques are then employed to consolidate the representations of objects based on the FBS similarity and connection strength among them.

The **primary contributions** of this paper are:

- A novel object consolidation approach, which, unlike traditional techniques, employs not only attribute (feature) similarity, but also analyzes inter-object relationships to improve the quality of consolidation (Section 4).
- Novel metrics to analyze the quality of the outcome (Section 4.3).
- An empirical evaluation of the proposed technique, that establishes that analyzing relationships is important for object consolidation (Section 5).

Next, in Section 2, we present a motivational example and then, in Section 3, we formalize the problem and introduce the notation necessary to explain the approach.

2. MOTIVATING EXAMPLE

In this section we use an instance of the "author matching" problem to illustrate that exploiting chains of relationships, that exist among entities, can improve the quality of object consolidation.

Consider a toy database consisting of the *author* and *publication* records shown in Figures 1 and 2. Assume that the publications are represented in the database using the attributes ⟨`id,title,authorRef1`, ..., `authorRefN`⟩, where `id` is the paper identifier, `title` is the paper title, and `authorRef`'s are the names of the authors of the paper. Suppose that the author information is stored in the form ⟨`id,authorName,affiliation`⟩, where `id` is the author identifier, `authorName` and `affiliation` are the author's name and affiliation.

[1]For example, two strings 'J. Smith' and 'John Smith', while not identical, are sufficiently similar to suggest that one can be the other and FBS techniques can detect that. It can also be known from the context that the mentioned 'J. Smith' works at MIT and 'John Smith' works at MIT, then FBS approaches can use this additional attribute (affiliation) and suggest that they are now more confident that the two representations refer to the same person.

⟨$P1$, Title1, **'John Smith'**, 'Alan White'⟩
⟨$P2$, Title2, 'Alan White', 'Mike Black'⟩
⟨$P3$, Title3, **'J. Smith'**, 'Mike Black'⟩
⟨$P4$, Title4, 'Tom Grey', **'John Smith'**⟩
⟨$P5$, Title5, 'Tom Grey', 'Kate Red'⟩

Figure 1: *Publication* records

⟨$A1$, **'John Smith'**, 'MIT'⟩
⟨$A2$, **'John Z. Smith'**, 'CMU'⟩
⟨$A3$, **'John Smith'**, 'Stanford'⟩
⟨$A4$, 'Alan White', 'MIT'⟩
⟨$A5$, 'Mike Black', 'NEC'⟩
⟨$A6$, 'Tom Grey', 'Intel'⟩
⟨$A7$, 'Kate Red', 'Stanford'⟩

Figure 2: *Author* records

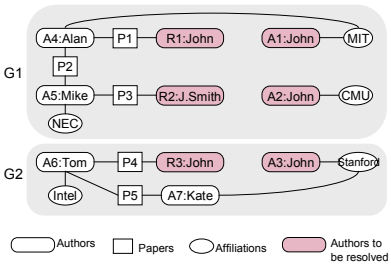


Figure 3: Graph for toy database.

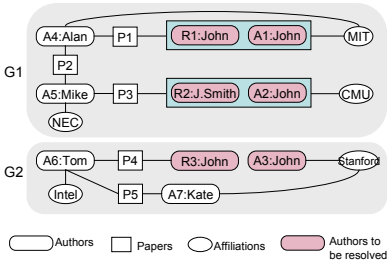


Figure 4: Adding context info.

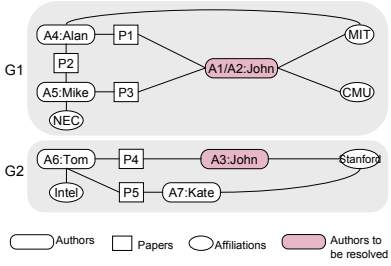


Figure 5: Adding relshp analysis.

We assume that in this database we are only uncertain about representations of *people*. For instance, we are uncertain to which author the representation 'John Smith' in the paper $P1$ refers to: A_1, A_2, or A_3. For that matter, we are also uncertain, whether A_1, A_2, or A_3 are representations of different people, or they are just duplicate records. The latter can be the case, for instance, if 'John Smith' was a graduate student at MIT and then became a faculty at CMU, so $A1$ and $A2$ are duplicates in this scenario. However, under our assumptions, we are certain that the representation 'MIT' in $A1$ is the same organization as the 'MIT' in $A4$, because these two representations are of the type *affiliation*, and not of the type *people*.

Traditional FBS techniques. To solve this problem using traditional techniques one would first try to deduplicate *author* records. After that, those records would be assumed to uniquely represent each distinct author and the goal would be to match `authorRef`'s in *publication* records to the correct authors. For example, existing feature-based similarity techniques can be used to compare the description in each `authorRef` in *publication* records with the values of the `authorName` attribute in *authors* records. Using this technique we can accurately consolidate most of the representations for our toy database, except for the 'John Smith' representations. For example, the author represented as 'Alan White' in publications $P1$ and $P2$ will be mapped uniquely to the author record $A4$ for 'Alan White'; 'Mike Black' in publications $P2$ and $P3$ will be correctly mapped to $A5$ and so on. The only difficulty will be with 'John Smith' and 'J. Smith' in $P1$, $P3$, and $P4$, since each of them can correspond to either $A1$, $A2$, or $A3$.

Let us note that we can visualize the resulting dataset as a graph. In this graph, each entity/object, as well as the not-yet-consolidated representations, become nodes. The relationships that exist among them are visualized as edges.

The graph for the toy database is illustrated in Figure 3. For instance, since author $A1$ is affiliated with 'MIT', there is an edge between them, that corresponds to this relationship. Since, under our assumptions, 'MIT' uniquely identifies the corresponding entity, only one node is created for all 'MIT' representations. However, each 'John Smith' has a separate node in this figure.

Employing context. The most recent data cleaning techniques, such as [7], are also capable of employing the *context* to improve the quality of cleaning. There might be additional context information ('context attributes') that such algorithms might be able to use.

For instance, if it is also known that the coauthor of $P1$, 'Alan White', is from 'MIT', then, given that $A1$ is from 'MIT' as well, we may decide that 'John Smith' in $P1$ refers to $A1$ and not to $A2$ or $A3$. As another example, if we already know that $A2$ has papers with titles similar to that of $P3$, then we can infer that 'J. Smith' in $P3$ refers to $A2$. The resulting dataset can be visualized as the graph illustrated in Figure 4, where $R1, A1$, and $R2, A2$ are shown to be merged into two nodes.

Analyzing relationships. Now we will show how additional semantic information, stored in the relationships that exist between entities, can help to improve the quality of cleaning even further.

Observation 1: ('John Smith' in $P1$). First, to handle author 'John Smith' in $P1$, we observe that his co-author 'Alan White' has also written a paper $P2$ with 'Mike Black', who in turn has a paper $P3$ with 'J. Smith'. This gives us certain evidence that 'John Smith' in $P1$ is the same person as 'J. Smith' in $P3$. The intuition behind it is that people in the similar/related research areas tend to cooperate with each other and form co-authorship networks. Based on this evidence, we might decide that $P1$ and $P3$ are written by the same author, whose name is 'John Smith'.

Recall, by using the context, we have determined above that 'John Smith' in $P1$ refers to $A1$, and 'J. Smith' in $P3$ to $A2$. Therefore, the evidence suggests that $A1$ and $A2$ are duplicate records for the same author – the fact that was not captured by the above feature-based similarity algorithm!

Observation 2: ('John Smith' in $P4$). Consider the task of deciding whether the representation 'John Smith' in $P4$ refers to $A1$, $A2$, or $A3$. Observe that the coauthor of that paper, 'Tom Grey', has a paper $P5$ with 'Kate Red', who is at 'Stanford'. The author $A3$ is also at 'Stanford', and thus we are able to establish a connection between 'John Smith' in $P4$ and $A3$. Given that there are no such connections to $A1$ an $A2$ that we can find, we might decide that 'John Smith' in $P4$ probably refers to $A3$.

Generic approach. At first glance, the analysis in Observations 1 and 2 might seem to be domain-specific. How-

ever, a domain-independent approach emerges if we view the underlying database as a graph of object representations (modeled as nodes) linked to each other via relationships (modeled as edges). The analysis can be viewed as application of the CAP hypothesis to the toy database, represented as the graph in Figure 4.

The first observation we made, regarding $R1$ and $R2$ being two representations for the same author, was based on the presence of the path (we use 'path', 'relationship chain', and 'connection' interchangeably):

$$R1 \rightarrow P1 \rightarrow A4 \rightarrow P2 \rightarrow A5 \rightarrow P3 \rightarrow R2.$$

Via this path, we were able to 'connect' $R1$ and $R2$.

The second observation we made regarding disambiguation of 'John Smith' in $P4$ was based on the presence of the path

$$R3 \rightarrow P4 \rightarrow A6 \rightarrow P5 \rightarrow A7 \rightarrow \text{'Stanford'} \rightarrow A3.$$

Via this path, we 'connected' $R3$ and $A3$.

Figure 5 shows the resulting graph, which reflects the outcome of the above analysis. In this figure, $R1$, $R2$, $A1$, and $A2$ are grouped together (forming the first group), since, those representations are likely to refer to the same person. Similarly, $R3$ and $A3$ are grouped as well, forming the second group. Let us observe that the graph in Figure 4 can be split into two connected subgraphs $G1$ and $G2$, such that $G1$ contains nodes of the first group and $G2$ of the second group.

In general, connections between representations can be more complex than those in this example. Therefore, a similar analysis may need to measure and compare the "strength" in the connections that exists between various entities.

Thus, the generic approach for object consolidation may consist of the following steps. First, the approach identifies the representations that can refer to the same entity. Then, it discovers *connections* between these representations and measures the connection strength between them to obtain the evidence to be used in the consolidation process. The algorithm then employs a graph partitioning algorithms to group the representations into clusters, such that the connections among the nodes in the same cluster are strong, and among the nodes across the clusters are weak, to satisfy the CAP principle.

Naturally, one should demonstrate that the CAP hypothesis holds over real datasets by designing a generic solution to exploiting relationships for object consolidation. We will develop one such general domain-independent strategy in Section 4. We perform an extensive study of the proposed approach over a real dataset, to establish that exploiting relationships can further improve the quality of object consolidation. Before we develop our solution, we first introduce the notation and concepts needed to explain the approach, in Section 3.

3. PROBLEM FORMULATION

3.1 Notation

Let $\mathcal{D}$ be the database being processed. We will use $O = \{o_1, o_2, \ldots, o_{|O|}\}$ to denote the set of all entities (or, *objects*) in $\mathcal{D}$. 'Entities' here have the same meaning as in the E/R model. Various consolidation scenarios are possible w.r.t. O. In one scenario, the consolidation algorithm has some information about the objects in O and about the cardinality of O. A more complicated scenario is when no information about the number or the nature of the objects in O is available.

Entities are referred in the database via *representations*. Let $X = \{x_1, x_2, \ldots, x_{|X|}\}$ (where $|X| \geq |O|$) be the set of all representations in $\mathcal{D}$. A representation is a description of an entity, which may consist of one or more attributes. For instance, in the toy database in Section 2, `authorRef` representations consist of only one attribute $\langle$*author name*$\rangle$. If, besides author names, author affiliation were also stored in the *publication* records, then `authorRef` references would have consisted of two attributes – $\langle$*author name*, *author affiliation*$\rangle$.

Each representation x_i semantically refers to a single specific entity in O, which we denote by $d[x_i]$, where $d[x_i]$, in general, is unknown to the consolidation algorithm. The **goal** is to group all the representations in X into a set C of $|O|$ non-empty clusters, $C = \{C_1, C_2, \ldots, C_{|O|}\}$, such that all the representations in one cluster refer to the same entity, and no two representations from two different clusters refer to the same entity.[2] That is, for any two representations x and y from the cluster C_i it should follow that $d[x] = d[y]$. Similarly, for any two representations x and y from two distinct clusters C_i and C_j, it should follow that $d[x] \neq d[y]$.

We will use $C[x_i]$ to denote the *group set* of x_i – the set of all the representations from X that refer to the same entity as x_i, and thus should be put into the same group with x_i: $C[x_i] = \{x_j \in X : d[x_j] = d[x_i]\}$. Similar to $d[x_i]$, the group set $C[x_i]$, in general, is unknown to the consolidation algorithm. Given this notation, the goal can be reformulated as determining $C[x_i]$ for each $x_i \in X$. Let $S[x_i]$ denote the *consolidation set* of x_i – the set of all representations from X such that x_i and any representation from $S[x_i]$ can potentially refer to the same entity based on their feature-based similarity. That is, $S[x_i] = \{x_j : sim(x_j, x_i) > \tau\}$, where sim denote a feature-based similarity function and τ is some threshold. We assume that $C[x_i] \subseteq S[x_i]$.

To illustrate these concepts, consider the database in Table 1. It contains four representations of people: x, x_A, x_B, and y. Assume that those representation are not misspelled, and therefore x_A and x_B cannot refer to the same person, i.e. $d[x_A] \neq d[x_B]$. Suppose that x and x_A are representations of one person, x_B is of another person, and y is of a third person. Then the corresponding cluster sets and consolidation sets are shown in Table 1.

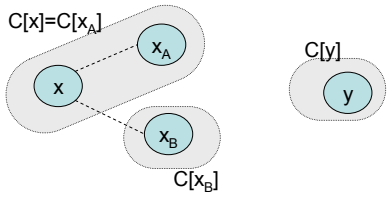


Figure 6: Similarity edges.

Figure 6 graphically illustrate this example. In this figure, a node is created per representation and a *similarity* edge is created between two nodes only if the corresponding

[2]Notice, the goal is only to be able to accurately group representations, **not** to infer any information about the entities they represent, etc.

Representations	C	S	S^*
$x =$ 'J. Smith'	$C[x] = \{x, x_A\}$	$S[x] = \{x, x_A, x_B\}$	$S^*[x] = \{x, x_A, x_B\}$
$x_A =$ 'J. A. Smith'	$C[x_A] = \{x, x_A\}$	$S[x_A] = \{x, x_A\}$	$S^*[x_A] = \{x, x_A, x_B\}$
$x_B =$ 'J. B. Smith'	$C[x_B] = \{x_B\}$	$S[x_B] = \{x, x_B\}$	$S^*[x_B] = \{x, x_A, x_B\}$
$y =$ 'Alan White'	$C[y] = \{y\}$	S[y] = {y}	$S^*[y] = \{y\}$

Table 1: Example to illustrate the notation.

representations may refer to the same entity, as determined by their feature-based similarity.

In general, such a *virtual similarity graph* can be created for the set X of all the representations in $\mathcal{D}$. Typically, this graph is composed of multiple *virtual connected subgraphs (VCS)*. To define the VCS for a representation x_i, we first define the set of all the representations $S^*[x_i]$ that belong to the same VCS as x_i. The set $S^*[x_i]$, consists of x_i and each representation $y \in X$, such that there exist a path $x_i \rightsquigarrow y$, consisting of only similarity edges. Notice that $C[x_i] \subseteq S[x_i] \subseteq S^*[x_i]$, as illustrated in Table 1. For instance, the graph in Figure 6 consists of two VCS's: one with nodes $\{x, x_A, x_B\}$ and the other one with node $\{y\}$.

The concept of a VCS is useful because the representations that belong to different VCS's cannot refer to the same entity. This allow us to cluster references in a "one VCS at a time" fashion.

3.2 The Attributed Relational Graph

Our consolidation approach views the database $\mathcal{D}$ as an undirected attributed relational graph (ARG) $G = (V, E)$, where V is the set of nodes and E is the set of edges. ARGs are often utilized by various applications to represent the entities (nodes) in a dataset, connected via relationships (edges). The graph is called *attributed* because attributes can be associated with both the edges and nodes of such a graph. We use ARGs in a similar manner, as elaborated below.

Nodes. In real-world datasets all the representations can be divided into two categories: those for which consolidation is trivial and those for which this task requires extra processing. For instance, in the toy database all representations of *affiliations* can be trivially consolidated by applying feature-based similarity since this similarity was sufficient to uniquely identify each distinct entity. However, consolidation of certain *author* representations required an additional processing. For those cases where consolidation is trivial, the representations of the same entity are clustered, and a node is created per the resulting cluster of representations. For those cases where consolidation is not trivial, a node is created per representation. Figure 4 illustrates the resulting graph for the toy database from Section 2. Let us note that the way our approach creates nodes closely resembles the way they are typically created in ARGs – to represent distinct entities.

Edges. The approach handles two types of edges: *regular* and *similarity* edges. Regular edges connect representations of entities if they are related via relationships.[3] For instance, in the graph in Figure 3 in Section 2, edges connect all the representations of authors and the papers they have written. A *similarity* edge is created for each pair of representation that can refer to the same entity (based on feature-based similarity). Each similarity edge has a weight associated with it (a real number from [0,1] interval) which reflects the degree of similarity between the two representations based on their features. Similarity edges for the toy database are illustrated in Figure 7.

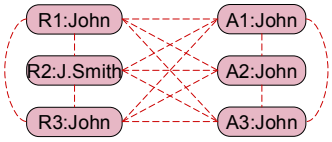


Figure 7: Virtual connected subgraph

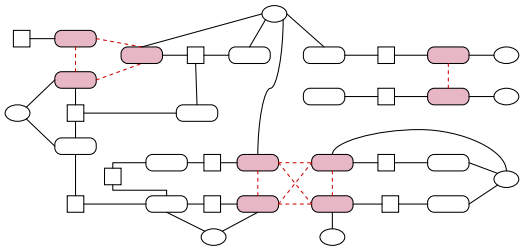


Figure 8: Graph Example

Representing nodes and edges graphically. We will use solid lines to graphically represent regular edges and broken lines for similarity edges. Nodes that correspond to already consolidated clusters of representations will not have color; representations that are yet to be consolidated are represented as shaded nodes. So an ARG might look like the one illustrated in Figure 8. Notice how the similarity edges define the three distinct VCS's in this example.

3.3 Connection Strength

In Section 1, we discussed that the approach consolidates representations based on the CAP hypothesis. To achieve that, it utilizes the notion of *connection strength* between two representations x and y, denoted as $c(x, y)$. This measure captures how strongly x and y are connected to each other via relationships. Many different models for computing $c(u, v)$ have been proposed in the literature, and we will take up one of them in Section 4.1.

4. THE CONSOLIDATION ALGORITHM

We now have developed all the concepts and notation needed to explain our approach for object consolidation. The approach exploits both features and relationships for the purpose of consolidation, and outputs the resulting clus-

[3]We will concentrate primarily on binary relationships. Multiway relationships are rare and most of them can be converted to binary relationships [11]. Most of the design models/tools only deal with binary relationships, for instance ODL (Object Definition Language) supports only binary relationships.

tering as the outcome. The approach consolidates representations using the following steps:

1. **Construct the ARG and identify all VCS's.** The first step is to construct the ARG for the dataset. We assume that feature-based similarity is used in constructing such a graph. This step has been explained in detail in Section 3.2.

2. **Choose a VCS and compute $c(u, v)$'s.** Pick a VCS in the ARG to be partitioned next. Then, compute the connection strength $c(u, v)$ for each pair of representations u and v in the VCS that are connected via a similarity edge.

3. **Partition the VCS.** Take, from Step 2, the VCS and the connection strength values. Use a graph partitioning algorithm to partition the VCS into clusters. The partitioning is carried out based on the connection strengths. After the VCS is partitioned, adjust the ARG accordingly. If the VCS was the last to be partitioned, then stop. Otherwise, go to Step 2.

We now discuss the above steps in more detail in the following subsections.

4.1 Computing Connection Strength

The connection strength measure $c(u, v)$ for two objects u and v computes how strongly they are connected to each other via relationships.

4.1.1 Existing models.

Recently, there has been a spike of interest by various research communities in the measures directly related to the $c(u, v)$ measure. Since the $c(u, v)$ measure is at the core of the proposed consolidation approach, we next analyze several principal existing models for computing $c(u, v)$.

Diffusion Kernels. The earliest work in this direction that we can trace is in the area of kernel-based pattern analysis [29]. The kernel methodology currently undergoes very active development, and shows a great promise for improving various pattern analysis tasks. In particular, this area studies 'diffusion kernels on graph nodes', which are of direct interest in our context and are defined as follows.

A *base similarity graph* $G = (S, E)$ for a dataset S is considered. The vertices in the graph are the data items in S. The undirected edges in this graph are labeled with a 'base' similarity $\tau(\mathbf{x}, \mathbf{y})$ measure. That measure is also denoted as $\tau_1(\mathbf{x}, \mathbf{y})$, because only the direct links (of size 1) between nodes are utilized to derive this similarity. The base similarity matrix $\mathbf{B} = \mathbf{B}_1$ is then defined as the matrix whose elements $\mathbf{B}_{\mathbf{xy}}$, indexed by data items, are computed as $\mathbf{B}_{\mathbf{xy}} = \tau(\mathbf{x}, \mathbf{y}) = \tau_1(\mathbf{x}, \mathbf{y})$. Next the concept of base similarity is naturally extended to path of arbitrary length k. To define $\tau_k(\mathbf{x}, \mathbf{y})$, the set of all paths $P^k_{\mathbf{xy}}$ of length k between the data items $\mathbf{x}$ and $\mathbf{y}$ is considered. The similarity is defined as the sum over all these paths of the products of the base similarities of their edges:

$$\tau_k(\mathbf{x}, \mathbf{y}) = \sum_{(\mathbf{x}_1 \mathbf{x}_2 \ldots \mathbf{x}_k) \in P^k_{\mathbf{xy}}} \prod_{i=1}^{k} \tau_1(\mathbf{x}_{i-1}, \mathbf{x}_i)$$

Given such $\tau_k(\mathbf{x}, \mathbf{y})$ measure, the corresponding similarity matrix $\mathbf{B}_k$ is defined. It can be shown that $\mathbf{B}_k = \mathbf{B}^k$. The idea behind this process is to enhance the base similarity by those indirect similarities. For example, the base similarity $\mathbf{B}_1$ can be enhanced with similarity $\mathbf{B}_2$, e.g. by considering a combination of the two matrices: $\mathbf{B}_1 + \mathbf{B}_2$. The idea generalizes to more then two matrices. For instance, by observing that in practice the relevance of longer paths should decay, it was proposed to introduce a decay factor λ and define what is known as the *exponential diffusion kernel*: $\mathbf{K} = \sum_{k=0}^{\infty} \frac{1}{k!} \lambda^k \mathbf{B}^k = \exp(\lambda \mathbf{B})$. The *von Neumann diffusion kernel* is defined similarly: $\mathbf{K} = \sum_{k=0}^{\infty} \lambda^k \mathbf{B}^k = (\mathbf{I} - \lambda \mathbf{B})^{-1}$. The diffusion kernels can be computed efficiently by performing eigen-decomposition of $\mathbf{B}$, that is $\mathbf{B} = \mathbf{V}' \mathbf{\Lambda} \mathbf{V}$, where the diagonal matrix $\mathbf{\Lambda}$ contains the eigenvalues of $\mathbf{B}$, and by making an observation that for any polynomial $p(x)$, the following holds $p(\mathbf{V}' \mathbf{\Lambda} \mathbf{V}) = \mathbf{V}' p(\mathbf{\Lambda}) \mathbf{V}$. The elements of the matrix $\mathbf{K}$ exactly define what we refer to as the connection strength: $c(\mathbf{x}, \mathbf{y}) = \mathbf{K}_{\mathbf{xy}}$.

The solutions proposed for the diffusion kernels work well, if the goal is to compute $c(u, v)$ for all the elements in the dataset. They are also very useful for illustration purposes and similar in nature (though cannot be used "as is") to the weight-based model we employ in our previous work [14, 17]. However, in data cleaning, the task is frequently to compute only some of $c(u, v)$'s, thus more efficient solutions are possible. Also, often after computing one $c(u, v)$, the graph is adjusted in some way, which affects the values of $c(u, v)$'s computed after that.

Relevant importance in graphs. White et al. in [33] consider the problem of computing 'relevant importance' of a set of nodes in a graph with respect to the set of 'root' nodes. The problem of computing $c(u, v)$ can be postulated as computing the relevant importance of node u with respect to the root node v. In [33] several known techniques are evaluated for their goal. Basic techniques, such as the length of the shortest paths between u and v, are compared against more involved ones, which are similar to the diffusion kernels. In the context of our problem, the work in [33] has an advantage over the work on kernels, because [33] focuses on efficient computation of one $c(u, v)$, whereas the kernel methods compute the whole similarity matrix $\mathbf{K}$.

Electric circuit analogy. Faloutsos et al. in [9] considers a model for computing $c(u, v)$. They view the graph as an electric circuit consisting of resistors, and compute $c(u, v)$ as the amount of electric current that goes from u to v. One of the primary contributions of that paper is the optimizations that scale their approach to large graphs.

Random walks in graphs. Another common model used for computing $c(u, v)$ is to compute it as the probability to reach node v from node u via random walks in the graph. That model has been studied extensively, including in our previous work [14, 17].

Problems with existing models. In the context of data cleaning, the existing techniques have several disadvantages. One disadvantage is that the true 'base' similarity is rarely known in real-world datasets. Some existing techniques try to mitigate that by *imposing* a similarity model. However, the CAP principle implies *its own* similarity measure, and any imposed model, created for its own sake in isolation from the specific application, might have little to do with it. Ideally, the similarity measure should be derived directly from data for the specific application at hand that employs it. One step toward achieving this, is to consider *parameterized* models and then try to learn an optimal

combination of parameters directly from data. We have explored such an approach in [16] for the problem of reference disambiguation. Next we will present the $c(u, v)$ model we use in this paper for object consolidation. That model is parameterized, however in this paper we do not study how to learn the right parameters, but rather assume they are assigned by the domain analyst, as we will elaborate shortly. Let us note that our overall approach is independent from a particular $c(u, v)$ model, and a different model, e.g. [17], can be utilized for this purpose.

4.1.2 The connection strength model.

The $c(u, v)$ model we use is very similar to that of the diffusion kernels. To compute $c(u, v)$, the set of all L-short simple paths $\mathcal{P}_L(u, v)$ between u and v is analyzed, where a path is L-short if its length does not exceed L. The total connection strength between nodes u and v is computed as the sum of connection strengths of paths in $\mathcal{P}_L(u, v)$:

$$c(u, v) = \sum_{p \in \mathcal{P}_L(u,v)} c(p). \quad (1)$$

The connection strength $c(p)$ of each individual path p is computed the same way as in the kernel formulae: as a product of base similarities of edges.

The differences between the proposed model $\mathcal{M}$ and that of the kernels $\mathcal{M}_K$ are as follows. The model $\mathcal{M}_K$ employs the decay factor λ, whereas $\mathcal{M}$ does not. In $\mathcal{M}_K$, we have $L = \infty$, in $\mathcal{M}$ the parameter L is finite, e.g. $L = 5$. In $\mathcal{M}_K$ all base similarities $\tau(u, v)$ are known, in $\mathcal{M}$ we will derive them in a particular way: the ultimate goal (not considered in this paper) is to eventually to be able to learn them directly from data.

The procedure for computation $c(u, v)$ consists of two logical phases. The first phase discovers connections/paths between u and v. The second phase computes/measures the strength in connections discovered by the first phase.

The connection discovery phase. In general there can be many connections between nodes u and v. Intuitively, many of those (e.g., very long ones) are not very important. To capture most important connections while still being efficient, the algorithm computes the set of all L-short simple paths $\mathcal{P}_L(u, v)$ between nodes u and v in graph G. This algorithm is the bottleneck of the overall approach. Several optimizations of this algorithms has been studied in [14,17], which achieve orders of magnitude in improvement. In this paper we employ the same optimizations.

The second phase computes the strength in the discovered connections using Equation (1). We are yet to specify how we compute the connection strength $c(p)$ of each individual path p from $\mathcal{P}_L(u, v)$ in Equation (1). Let us address this issue.

Motivating $c(p)$ formula. Which factors should be taken into account when computing the connection strength $c(p)$ of each individual path p? Figure 9 illustrates two different paths (or connections) between nodes u and v: $p_a = u \to a \to v$ and $p_b = u \to b \to v$. Let us understand which connection is better.

Both connections have the same length of two. One connection is going via node a and the other via node b. The intent of Figure 9 is to show that node b "connects" many nodes, not just u and v, whereas node a "connects" only u and v. For instance, in the context of the author matching problem u and v can be two authors, a can be a publication

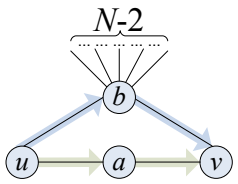


Figure 9: Motivating $c(p)$.

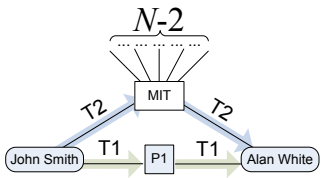


Figure 10: $c(p)$.

Figure 11: Experiments

and b a university, as illustrated in Figure 10. We argue the connection between u and v via b is much weaker than the connection between u and v via a: since b connects many nodes, it is not surprising we can connect u and v via b as well. Notice, measures such as path length, network flow do not capture the fact that $c(p_a) > c(p_b)$.

We compute $c(p)$ as follows. Let us assume first that path p consists of only regular edges, and no similarity edges. All edges in the ARG can be classified into a finite set of types $T = \{T_1, T_2, \ldots, T_m\}$. For example, for the author matching problem, type T_1 can be the edges that connect authors and publications, type T_2 can be the edges that connect authors and their affiliated organizations. So path p_a in Figure 10 can be viewed as a $\langle T_1, T_1 \rangle$ path and path p_b as a $\langle T_2, T_2 \rangle$ path. Each edge type T_i has a weight w_i (a real number from [0,1] interval) associated with it. The connection strength of path p is computed as the product of weights associated with its edge types. For example, if path p contains n_1 edges of type T_1, n_2 edges of type T_2 and so on, then $c(p)$ is computed as

$$c(p) = w_1^{n_1} w_2^{n_2} \times \cdots \times w_m^{n_m}. \quad (2)$$

Assigning weights. An important question is how those w_i weights are determined. There are several methods to accomplish that, we will briefly discuss only two methods:

1. The weights are assigned by the domain analyst.
2. The weights are learned from data using a supervised learning algorithm.

The first case is straightforward: the domain analyst, who is well-familiar with the dataset and the nature of this algorithm, picks the appropriate weights. However, it is often desirable to minimize the participation of the analyst. Thus, in our ongoing work, e.g. [16], we address the second case, where the challenge is to learn the weights automatically, directly from training data, by employing a supervised learning algorithm. Ideally, that solution should lead to a self-tunable algorithm, which achieves the best quality of consolidation.

Example. Without loss of generality, let us assume the weights are assigned by a domain analyst. Then, in the context of our motivating example, taking into account the fact that connections via publications are more unique than those via universities, the analyst might decide that the following combination of weights is reasonable: $w_1 = \frac{1}{2}$ and $w_2 = \frac{1}{10}$. So $c(p_a) = w_1^2 = \frac{1}{4}$, $c(p_b) = w_2^2 = \frac{1}{100}$, and $c(p_a) > c(p_b)$.

Paths with similarity edges. Let us consider paths that can contain similarity edges. Recall that a similarity edge between nodes for two representation x and y denote the fact that there is a chance that x and y can refer to the same

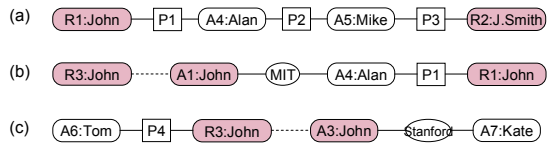


Figure 12: Paths with/without similarity edges.

entity. This edge has a weight w_F associated with it, which reflects the degree of similarity between x and y based on their features: $w_F = sim(x, y)$. We will refer to this weight as the FBS weight.

Let us note that a path containing a similarity edge might not even exist in reality. For instance, in Figure 12(a) the path consists of only regular edges and we are confident it exists. However, the existence of the path in Figures 12(b) depends on whether $R3$ and $A1$ refer to the same entity or not. The same applies to $R3$ and $A3$ in Figure 12(c).

The simplest solution is not to consider paths that contain similarity edges at all. Another (heuristic) solution we use is to associate the same very small *base* weight w_ε for all similarity edges. Then, compute the total weight v_i of a similarity edge as a product of its FBS weight w_F and the base weight w_ε: $v_i = w_F \times w_\varepsilon$. So, if path p has n_i edges of type T_i $(i = 1, 2, \ldots, m)$ and k similarity edges with total weights $v_1, v_2, \ldots, v_k$, then $c(p)$ is computed as

$$c(p) = w_1^{n_1} w_2^{n_2} \times \cdots \times w_m^{n_m} v_1 v_2 \times \cdots \times v_k. \qquad (3)$$

4.2 Consolidating objects by partitioning VCS's

To consolidate objects we need to partition the representations in each VCS. Each VCS contains representations of at least one object. If a representation of an object is contained in a VCS, then, by construction of VCSs, the rest of the representations of the same objects are contained in the same VCS. Two scenarios are possible. In one scenario, the knowledge of the number of objects contained in each VCS is available. In the other scenario, no such knowledge is available.

Assume we know that a given VCS contains the representations of exactly k objects. Then, the VCS must be partitioned into exactly k clusters. We need to consider all possible partitions of the VCS into k clusters, which are feasible according to the similarity edges in the VCS. There might be several such partitions and we should choose the one that best satisfies the CAP principle. We try to achieve that by employing a min-cut algorithm proposed in [30]. The standard min-cut problem is defined as follows. Given a weighted graph $G = (V, E)$, the goal is to partition V into two non-empty disjoint subsets V_1 and V_2, such that the total weight of the edges connecting the two parts, called the *cut* (i.e., the weight of $\{(u, v) \in E : u \in V_1; v \in V_2\}$), is minimized.

In our case, we partition V into not 2, but k subsets. The min-cut algorithm we employ [30] has two useful properties: (a) it utilizes a 'normalized' (to $[0, 2]$ interval) cut; and (b) it achieves the goal of minimizing connections across clusters and maximizing connections inside clusters at the same time. The former property will later allow us to specify a single threshold per all VCS's, while the latter property is important to better satisfy the CAP principle.

Defining weights for partitioning. A min-cut algorithm uses weights $w(u, v)$ between nodes for partitioning. The weights are defined as follows: if there is a similarity edge between u and v then $w(u, v) = c(u, v)$, otherwise $w(u, v) = 0$. Let us note that the formula for $w(u, v)$ can also include the feature-based similarity $sim(u, v)$, e.g. as a weighted sum: $w(u, v) = \alpha \times c(u, v) + (1 - \alpha) \times sim(u, v)$, $0 \leq \alpha \leq 1$. But we do not study this approach in this paper.

Further partitioning. Consider now the second scenario, where we do not know the number of objects in a given VCS. The algorithms handle this case by first partitioning VCS into two parts. Then, the algorithm decides whether to actually split the nodes into two clusters, or not. It achieves that by comparing the value of the resulting normalized cut c against predefined threshold τ. Currently, the value of τ is not yet learned from data, but rather is set by the domain analyst. If $c > \tau$, the two parts are still well inter-connected and the algorithm does not divide VCS further into two clusters. That means the algorithm assumes that all the representations in this VCS refer to a single real-world entity, and hence they should be grouped together in one cluster. On the other hand, if $c < \tau$, the algorithm repeatedly partition the resulting subgraphs until $c > \tau$. The resulting clustering of representations is returned as the final result of the algorithm.

4.3 Measuring the quality of outcome

The goal of object consolidation algorithms is to accurately group the representations of entities. However, consolidation algorithms can make mistakes, and thus the quality of the outcome should be quantified. Measures known as *dispersion* and *diversity* has been proposed before for this purpose [2].

Dispersion. We want the representations of the same entity to be clustered together in one cluster. The *dispersion* of a given entity captures the number of distinct clusters into which its representations are clustered. Therefore, the lesser the dispersion the better and the ideal dispersion is 1 for a given entity.

Diversity. We also want each cluster to contain representations of just one entity. The *diversity* of a given cluster captures the number of distinct entities whose representations are in this cluster. Similarly, the lesser the diversity the better and the ideal diversity is 1 for a given cluster.

Problems with dispersion and diversity. The *dispersion* and *diversity* do not always accurately reflect the quality of the outcome of object consolidation, although they are simple and easy to understand. Consider the following example. Assume a VCS to be partitioned is composed of $2n$ representations $a_1, a_2, \cdots, a_{2n}$ of entity E_A and of $2n$ representations $b_1, b_2, \cdots, b_{2n}$ of entity E_B. The goal is to group them correctly and therefore the ideal result is the two clusters:

$$C_1 = \{a_1, a_2, \ldots, a_{2n}\},$$
$$C_2 = \{b_1, b_2, \ldots, b_{2n}\}.$$

The algorithm however can make mistakes and the resulting two clusters might be:

$$C_1 = \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n\},$$
$$C_2 = \{a_{n+1}, a_{n+2}, \ldots, a_{2n}, b_{n+1}, b_{n+2}, \ldots, b_{2n}\},$$

that is, half of the representations are misassigned. In another situation, the two clusters might be

$$C_1 = \{a_1, a_2, \ldots, a_{2n-1}, b_{2n}\},$$

$$C_2 = \{b_1, b_2, \ldots, b_{2n-1}, a_{2n}\},$$

that is, just one is misassigned for each cluster.

Obviously, the latter answer is better than the previous one. However, in both situations, since the representations of both E_A and E_B are scattered across two clusters, the *dispersion* of each of the two entities, E_A and E_B, is 2. The *diversity* of each of the two clusters C_1 and C_2 is 2, since each cluster contains the representations of both E_A and E_B. Therefore, the *dispersion* and *diversity* measures cannot distinguish the two situations in this case.

Entropy-based quality measures. We argue that the metrics known as *entropy* can be utilized to better capture the above situations. Entropy was first proposed by Shannon in his famous work [28] on the mathematical theory of communication.

The *entropy* of a discrete random variable X reflects the degree of uncertainty associated with possible values of X. Let variable X take values from set $\{x_1, x_2, \ldots, x_n\}$ with the respective probabilities $p(x_1), p(x_2), \ldots, p(x_n)$, where $p(x_i) \neq 0$ and $p(x_1) + p(x_2) + \cdots + p(x_n) = 1$. Then the entropy of X, denoted $H(X)$, is defined as:

$$H(X) = \sum_{i=1}^{n} p(x_i) \, log_2 \frac{1}{p(x_i)}$$

Entropy $H(X)$ attains its minimum value $H(X) = 0$, when there exists some i such that $p(x_i) = 1$. In that situation there is no uncertainty associated with X: the value of X is always x_i. On the other hand, $H(X)$ attains its maximum value in the most uncertain scenario: when all the values $x_1, x_2, \ldots, x_n$ are equally likely, in which case $H(X) = log_2 \, n$. Thus, $H(X) \in [0, log_2 \, n]$.

Entity entropy. Assume that a certain entity E has m representations in the database, which are assigned to n clusters $C_1, C_2, \ldots, C_n$ by the algorithm. The assignment is such that m_1 representations are assigned to the cluster C_1, m_2 to C_2 and so on, such that $m_i \neq 0$ and $m_1 + m_2 + \cdots + m_n = m$. To measure the spread of representations of E over the clusters, we consider the fractions of all representations of E assigned to each cluster C_i $(i = 1, 2, \ldots, n)$: $p_i = \frac{m_i}{m}$. Then, using those fractions, we utilize entropy to quantify the spread of the entity's representations: $H(E) = \sum_{i=1}^{n} p_i \, log_2 \frac{1}{p_i}$. Let us note that the dispersion for this case is always n. The lower the value of $H(E)$, the better. The ideal value is 0.

Cluster entropy. Similarly, we can define the cluster entropy. Assume that the algorithm assigns to a cluster C exactly m representations, that correspond to n entities. Suppose that m_1 of them are representations of entity E_1, m_2 of E_2, ..., m_n of E_n, where $m_i \neq 0$ and $m_1 + m_2 + \cdots + m_n = m$. Like in the case above, we consider fractions $p_i = \frac{m_i}{m}$ and then use them in entropy to quantify the spread of the cluster's representations: $H(C) = \sum_{i=1}^{n} p_i \, log_2 \frac{1}{p_i}$. Let us observe that the diversity in this case is always n. The lower the value of $H(C)$ the better, the ideal value is 0.

Example. Consider the example above with the two clusters C_1 and C_2, and assume that $n = 10$. In the first situation, the entity entropy is:

$$H(E_A) = \tfrac{n}{2n} log_2 \tfrac{1}{n/2n} + \tfrac{n}{2n} log_2 \tfrac{1}{n/2n} = 1,$$
$$H(E_B) = 1,$$
$$avg = \tfrac{H(E_A)+H(E_B)}{2} = 1.$$

The cluster entropy is:

$$H(C_1) = \tfrac{n}{2n} log_2 \tfrac{1}{n/2n} + \tfrac{n}{2n} log_2 \tfrac{1}{n/2n} = 1,$$
$$H(C_2) = 1,$$
$$avg = \tfrac{H(C_1)+H(C_2)}{2} = 1.$$

In the second situation, the entity entropy is:

$$H(E_A) = \tfrac{2n-1}{2n} log_2 \tfrac{1}{(2n-1)/2n} + \tfrac{1}{2n} log_2 \tfrac{1}{1/2n} = 0.0703,$$
$$H(E_B) = 0.0703,$$
$$avg = \tfrac{H(E_A)+H(E_B)}{2} = 0.0703.$$

The cluster entropy is:

$$H(C_1) = \tfrac{2n-1}{2n} log_2 \tfrac{1}{(2n-1)/2n} + \tfrac{1}{2n} log_2 \tfrac{1}{1/2n} = 0.0703,$$
$$H(C_2) = 0.0703,$$
$$avg = \tfrac{H(C_1)+H(C_2)}{2} = 0.0703.$$

Let us observe that in contrast to the diversity and dispersion, the entropy-based measures do capture that the second partitioning is better than the first one, since $0.0703 < 1.0$.

5. EXPERIMENTAL EVALUATION

In this section, we experimentally study the proposed approach on a real dataset. We conducted the experiments on a 2GHz Pentium 4 machine with 1GB RAM. In the rest of this section, we first describe the dataset we use and then present the experiments that test the quality and the efficiency of the proposed technique.

5.1 RealMov Dataset

One of the dataset used in data cleaning research is the movies dataset, available from [34]. It is a real public-domain dataset. In this paper we refer to a processed version of it as 'RealMov'. RealMov contains entities of three types: *movies* (11,453 entities), *studios* (992 entities), and *people* (22,121 entities), which are stored in *movies*, *studios* and *people* tables respectively. The *movies* table contains multiple attributes, such as the title of a movie, the director, the producer, the studio producing the movie, the studio distributing the movie, etc. The *studios* table has attributes such as studio name, the founder of the studio, the year of foundation, etc. The *people* table contains attributes like a person's name, the date of birth, gender, etc. There is also a *cast* table storing all the actors of each movie.

This dataset contains five types of (regular) relationships: *movie_actor*, *movie_director*, *movie_producer*, *producingStudio*, and *distributingStudio*, which map movies to their actors, directors, producers, producing studios and distributing studios respectively. Figure 13 presents a sample graph for RealMov dataset. Relationships *movie_actor*, *movie_director*, and *movie_producer* connect entities of type *movies* to entities of type *people*. Relationships *producingStudio* and *distributingStudio* connect entities of types *movies* and *studios*.

A clean version of this dataset is available, in which all entities and relationships among them are accurately captured. This will allow us to test the quality of various consolidation techniques: by comparing their output against the true situation. Figure 13 shows a sample ARG for RealMov dataset. Each entity is represented as a node, and each relationship as an edge.

5.2 Quality experiments

Typically, two aspects of data cleaning algorithm are evaluated empirically: the quality of their outcome and their

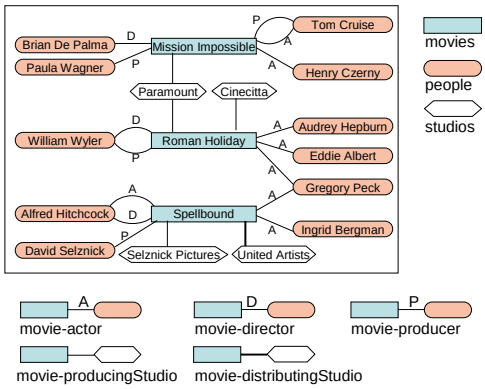


Figure 13: Graph example for movie dataset

efficiency. In this section, we study the quality of our approach on consolidating the representations of *director* entities. Let us note that the true mapping between the movies and their directors is available. Having this knowledge is the advantage of this dataset, since it will allow to compute the quality of various consolidation techniques.

Constructing experiment data. To test our approach, we will use a standard technique, commonly employed by data cleaning practitioners, e.g. in [4]: we introduce uncertainty/errors (in director representations) manually. The uncertainty will be introduced differently in different experiments, as explained next.

Assume that RealMov stores information about $d_1, d_2, \ldots, d_n$ director entities. We first choose randomly a fraction ρ of those directors: all their representations will be made uncertain. Based on the typical degree of uncertainty in real-world dataset [17], we set ρ to either 1%, 5%, 10%, 15%. Suppose that, say, the directors d_1, d_2, ..., d_{10} were chosen. We will make uncertain all the representations that refer to them, whereas for the rest of the directors $d_{11}, d_{12}, \ldots, d_n$, all their representations will still uniquely identify the right director.

To achieve that, we group the "uncertain" directors d_1, d_2, ..., d_{10} in some fashion, say in groups of two, e.g. $\{d_1, d_2\}$, $\{d_3, d_4\}$, ..., $\{d_9, d_{10}\}$. Then, we simulate the desired FBS uncertainty by changing all the representations of directors that belong to one group, such that each representation can refer to all directors in this group – if only the FBS similarity is utilized. For instance, assume that the dataset contains a representation r_1, which could only represent d_1. We modify r_1 such that it will fit the description of both, d_1 and d_2, – but will not fit the description of any other director. Such a constructed dataset is characterized by two types of parameters: ρ and the sizes of those groups.

Baseline methods. To reflect how our approach would compare against FBS techniques, we construct two baseline methods: *Baseline 1* and *Baseline 2*. Recall that our algorithm is applied only to 'tough' cases – after the existing FBS methods have already been used to successfully consolidate many of the representations (e.g., those "certain" representations). We now only test the quality of consolidating those 'tough' cases, which cannot be disambiguated by FBS methods.

Given that FBS cannot be used further to distinguish between the representations, whereas we would like to compare our algorithm against at least simple solutions, we construct our baseline methods as follows.

- *Baseline 1* method creates one cluster per each VCS and then assigns all the VCS's representations to this one cluster. That is, that method does not partition the VCS's further. This naïve method always achieves the ideal dispersion and entity entropy, because entities end up in just one cluster, as they should. However, if the VCS contains the representation of m objects, then the diversity of the cluster will be m, whereas the ideal diversity is 1.

- *Baseline 2* method knows the statistics of how many director groups there are of size 2, of size 3, and so on. Based on this statistics, for a given VCS, it first selects the number of partitions to split this VCS into. It then creates that many clusters, and randomly assigns representations from the VCS to each cluster.

Experiment 1. In this experiment, we set $\rho = 1\%$ and the size of each group of directors to be 2. The results for $\rho = 5\%$, 10% and 15% closely resemble those for $\rho = 1\%$ and thus omitted. For this experiment we also make our algorithm (and Baseline 2) aware that each resulting VCS must be partitioned into exactly two clusters.

Figure 14(c) and 14(d) show the average diversity and dispersion as we vary parameter L . Recall that we consider only L-short paths, or paths of length no greater than L. The results show that additional semantic information, stored in inter-object relationship, improves the quality of object consolidation. They also show that longer paths help improve the quality of the outcome.

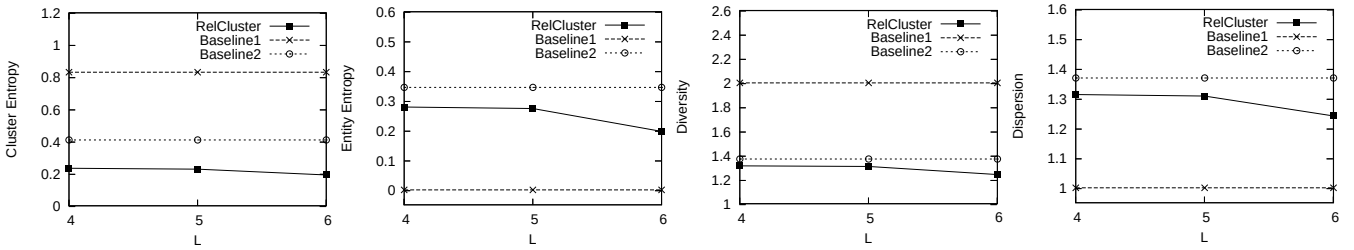
Since Baseline 1 does not partition VCS's at all, each resulting cluster always contains representations of 2 entities, and thus the diversity is always 2. Also, since the nodes of the same entity are always grouped into the same cluster, the entity dispersion is always 1 (the ideal dispersion).

Compared to the diversity and dispersion, the entropy should more properly capture the composition of groups. Figures 14(a) and 14(b) show the average cluster- and entity-entropy, achieved by the consolidation approaches. Recall that the lower the entropy, the better, and that the ideal entropy is zero. The figures look similar to the figures for the diversity and dispersion.

Experiment 2. In this experiment, the size of each director group is not 2 as in the previous experiment, but chosen randomly as 2, 3, or 4 with equal probability. Also our approach now is not aware into how many clusters each VCS should be clustered but rather utilizes threshold τ to decide that, as discussed in Section 4.2.

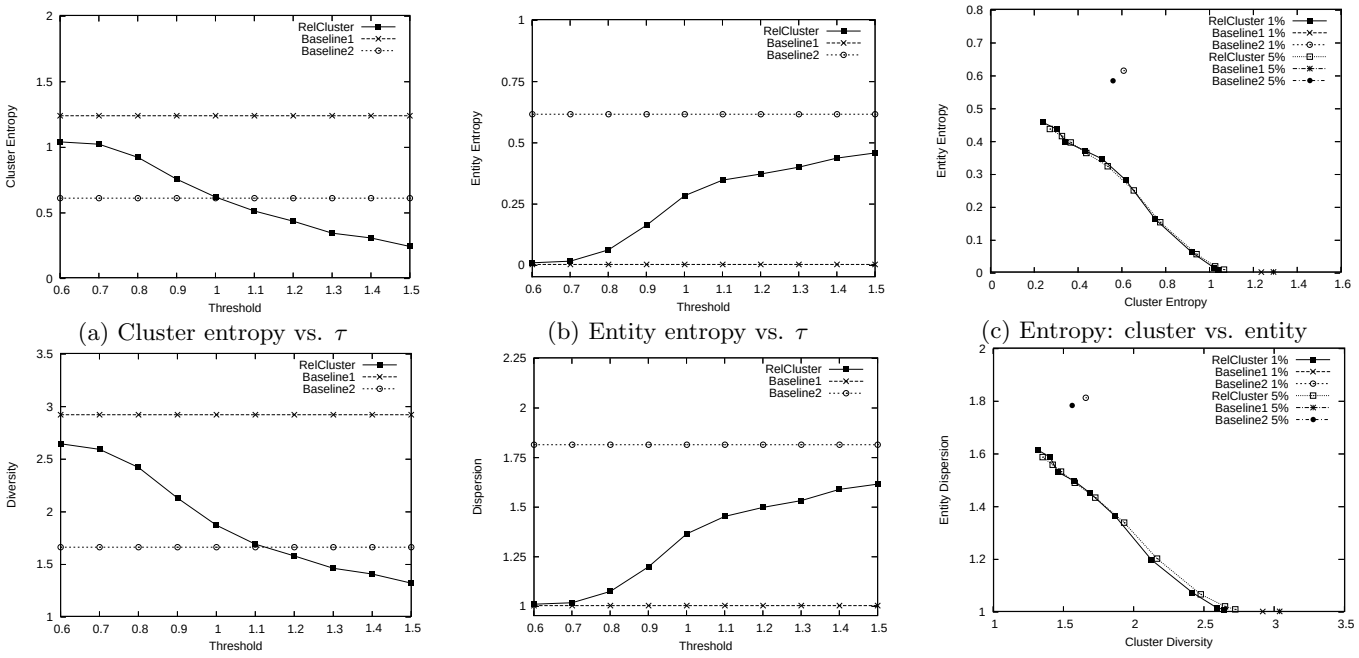
Figure 15 studies the effects of τ on the quality of the output. When τ is small, the normalized cut of most partitions is greater than τ, so the further partitioning is not carried out. Therefore, representations in each VCS are likely to be grouped into a small number of clusters and that is why the results closely resemble those of Baseline 1. On the other hand, large τ leads to creating many clusters for each VCS. This improves the cluster entropy, but the entity entropy becomes worse. So, there is a natural trade-off between the cluster entropy and entity entropy.

Figure 15(c) plots the cluster and entity entropy in one figure. Such a figure is useful for the analysts to pick the right value of τ, such that the desired compromise between the values of cluster- and entity- entropies is achieved. Figures 15(d), 15(e), 15(f) are similar to Figures 15(a), 15(b), 15(c), but for the diversity and dispersion.



(a) Cluster entropy (b) Entity entropy (c) Cluster diversity (d) Entity dispersion

Figure 14: Experiments with various lengths of paths



(a) Cluster entropy vs. τ (b) Entity entropy vs. τ (c) Entropy: cluster vs. entity

(d) Cluster diversity vs. τ (e) Entity dispersion vs. τ (f) Diversity vs. Dispersion

Figure 15: Experiments with various thresholds

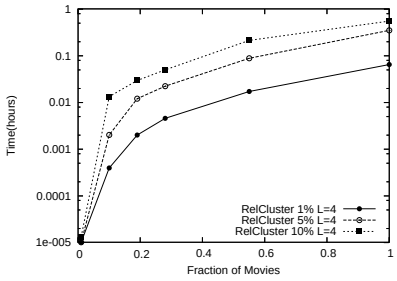


Figure 16: Execution time vs. database size.

5.3 Efficiency

Experiment 3. This experiment tests the efficiency of the proposed approach. Figure 16 shows the execution time of RelCluster as a function of the fraction of movies from RealMov dataset, e.g. 1.0 corresponds to the whole RealMov dataset. The bottleneck of our approach is the algorithm for discovering all L-short simple paths. In [14] we study several optimizations of that algorithm, which improve the performance by 1–2 orders of magnitude. We employ the same optimizations in out implementation of RelCluster.

6. RELATED WORK

Many research challenges have been explored in the context of data cleaning: dealing with missing data, handling erroneous data, record linkage, and so on. The closest to the problem of object consolidation addressed in this paper is the problem of record linkage. The importance of record linkage is underscored by the large number of companies, such as Trillium, Vality, FirstLogic, DataFlux, which have developed domain-specific record linkage solutions.

Researchers have also explored domain-independent techniques, e.g. [1, 10, 13, 19, 23]. Their work can be viewed as addressing two challenges: (1) improving similarity function, as in [3]; and (2) improving efficiency of linkage, as in [4]. Typically, two-level similarity functions are employed to compare two records. First, such a function computes attribute-level similarities by comparing values in the same attributes of two records. Next, the function combines the attribute-level similarity measures to compute the overall similarity of two records. A recent trend has been to employ machine learning techniques, e.g. SVM, to learn the best similarity function for a given domain [3]. Many techniques have been proposed to address the efficiency challenge

as well: e.g. using specialized indexes [4], sortings, etc.

Those domain-independent techniques deal only with attributes. Only one existing approach [17] analyzes relationships in a fashion similar to that proposed in this paper. That approach, and the one proposed in this paper, are part of the *Relationship-based Data Cleaning (RelDC)* project [15] at UCI. However, [17] solves a different data cleaning challenge, called reference disambiguation. That problem is known to be a subproblem of the problem of object consolidation and the approach proposed in [17], in general, cannot be used to solve the problem addressed in this paper. That approach converts the cleaning task to solving a nonlinear programming problem whereas we employ partitioning techniques for data cleaning. Other researchers have also proposed using relationships for cleaning, but in a different fashion. In [1] Ananthakrishna et al. employ similarity of directly linked entities, for the case of hierarchical relationships, to solve the record deduplication challenge. In [18] Lee et al. develop an association-rules mining based method to disambiguate references using similarity of the context attributes: the proposed technique is still an FBS method, but [18] also discusses "concept hierarchies" which are related to relationships. Getoor et al. in [2] use similarity of attributes of directly linked objects, like in [1], for the purpose of object consolidation. However, applying that technique in practice on real-world datasets was identified as future work in that paper. In contrast to the above described techniques, our approach and [17] utilize the CAP hypothesis to automatically discover and analyze relationship chains, thereby establishing a framework that employs systematic relationship analysis for the purpose of cleaning.

7. CONCLUSION

In this paper, we have shown that analysis of inter-object relationships is important for object consolidation and have demonstrated one approach that utilizes relationships for this purpose. Our ongoing work [16] addresses the challenge of automatically adapting the proposed data cleaning techniques to datasets at hand, by learning how to weigh different connections directly from data, in an automated fashion. Solving this challenge, in general, not only makes the approach a plug-and-play solution, but also improves both the accuracy and efficiency of the approach as discussed in [16].

8. REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.

[2] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *DMKD Workshop*, 2004.

[3] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD'03*.

[4] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD Conf.*, 2003.

[5] P. Christen, T. Churches, and J. X. Zhu. Probabilistic name and address cleaning and standardisation. The Australasian Data Mining Wshp, 2002.

[6] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *SIGKDD*, 2002.

[7] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.

[8] M. G. Elfeky and V. S. Verykios. On search enhancement of the record linkage process. In *KDD-2003 Wshp on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[9] C. Faloutsos, K. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *SIGKDD*, 2004.

[10] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[11] H. Garcia-Molina, J. Ullman, and J. Widom. *Database systems: the complete book*. Prentice Hall, 2002.

[12] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB01*.

[13] M. Hernandez and S. Stolfo. The merge/purge prob- lem for large databases. In *SIGMOD*, 1995.

[14] D. Kalashnikov and S. Mehrotra. Exploiting relationships for domain-independent data cleaning. *SIAM SDM*, 2005. ext. ver., `http://www.ics.uci.edu/~dvk/pub/sdm05.pdf`.

[15] D. V. Kalashnikov and S. Mehrotra. RelDC project. `http://www.ics.uci.edu/~dvk/RelDC/`.

[16] D. V. Kalashnikov and S. Mehrotra. Learning importance of relationships for reference disambiguation. *UCI Technical Report RESCUE-04-23*, Dec. 2004. `http://www.ics.uci.edu/~dvk/RelDC/TR/TR-RESCUE-04-23.pdf`.

[17] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM International Conference on Data Mining (SIAM SDM 2005)*, Newport Beach, CA, USA, April 21–23 2005.

[18] M. Lee, W. Hsu, and V. Kothari. Cleaning the spurious links in data. *IEEE Intelligent Systems*, Mar-Apr 2004.

[19] A. K. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *SIGKDD*, 2000.

[20] M. Michalowski, S. Thakkar, and C. Knoblock. Exploiting secondary sources for automatic object consolidation. In *KDD-2003 Wshp on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[21] A. E. Monge and C. P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *SIGMOD Wshp on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[22] M. Neiling and S. Jurk. The object identification framework. In *KDD-2003 Wshp on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[23] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[24] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In Advances in Neural Processing Systems 15. Vancouver, British Columbia:MIT Press, 2002.

[25] D. Quass and P. Starkey. Record linkage for genealogical databases. In *KDD-2003 Wshp on Data Cleaning*, 2003.

[26] L. D. Raedt and et al. Three companions for data mining in first order logic. In *Relational Data Mining*. Springer-Verlag, 2001.

[27] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *SIGKDD*, 2002.

[28] C. E. Shannon. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[29] J. Shawe-Taylor and N. Cristianni. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[30] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[31] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string tranformation weights for high accuracy object identification. In *SIGKDD*, 2002.

[32] V. Verykios, G.V.Moustakides, and M. Elfeky. A bayesian decision model for cost optimal record matching. *The VLDB Journal*, 12:28–40, 2003.

[33] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *SIGKDD*, 2003.

[34] G. Wiederhold. `www-db.stanford.edu/pub/movies/`.

Blocking-Aware Private Record Linkage

Ali Al-Lawati
Penn State / CSE
allawati@cse.psu.edu

Dongwon Lee
Penn State / IST
dongwon@psu.edu

Patrick McDaniel
Penn State / CSE
mcdaniel@cse.psu.edu

ABSTRACT

In this paper, the problem of quickly matching records (i.e., record linkage problem) from two autonomous sources without revealing privacy to the other parties is considered. In particular, our focus is to devise *secure blocking* scheme to improve the performance of record linkage significantly while being secure. Although there have been works on private record linkage, none has considered adopting the blocking framework. Therefore, our proposed blocking-aware private record linkage can perform large-scale record linkage without revealing privacy. Preliminary experimental results showing the potential of the proposal are reported.

1. INTRODUCTION

The task of integrating similar databases populated at separate locations to improve data qualities and enable accurate data analysis is often restricted by heterogeneity in the data. Specifications of how data is represented differ across databases of different parties. For example, "penn state university" can appear as simply "penn state", or as "The Pennsylvania State University". The goal of *record linkage* [16] is to identify similar records with precision, thereby facilitating accurate pattern and data analysis. In record linkage, all data to be matched appears in its original form in the matching process. This is acceptable when the data is of little value or when participants are mutually trusting. However, it is inappropriate to reveal privacy of autonomous sources. The goal of *private record linkage* [7] is, thus, to identify similar records without revealing privacy to others [1]. Applications that demand such a private record linkage include the medical field where patient records must be shared among hospitals and institutions while the identity of the patients are sealed.

Given two data sources, X and Y, the most naive form of record linkage is to perform pair-wise comparison – each record x from X and y from Y are compared one by one, having a quadratic time complexity of $O(|X||Y|)$. Furthermore, each record x and y are typically examined by some distance metrics – if $dist(x, y)$ exceeds some threshold then the pair is "matched". One of the popular techniques to improve the performance of record linkage methods is to use *blocking* – by clustering records into pre-determined blocks so that expensive distance measures are performed to only records within each block. Typical blocking works as follows: Suppose one pre-groups records in Y into Y/b blocks (i.e., each block has b records on average). Then, each record x from X is compared to only records from one block and the matching record is determined. That is, the time complexity is reduced to $O(|X||b|)$. Since $b \ll X$, this blocking in general improves the performance.

As the data size grows and more expensive distance metrics are employed, the importance of blocking increases as well. However, in the case of private record linkage, to our best knowledge, no previous approaches attempt to combine the "blocking" with the "private record linkage." Therefore, in this paper, we study a *blocking-aware private record linkage* protocol and propose several blocking schemes for enhanced performance. A key observation is the preprocessing of records into blocks, using "secure blocking" schemes.

Example 1 (Motivation). To illustrate the benefits of blocking-aware private record linkage, consider two credit card companies that wish to identify fraudulent customer list common to both companies. However, using the regular record linkage is inappropriate because of the private nature of credit card information (i.e., two companies do not want to share their customer-related information with the other party). Assume company A holds 2 million records and company B holds 3 million records. Moreover, suppose an average of 50 records are assigned to each block. In the absence of blocking, every record pair has to be compared, resulting in 2 million $\times$ 3 million = 6 trillion comparisons! However, blocking reduces the number of comparisons to 2 million $\times$ 50 = 100 million. □

2. RELATED WORK

By and large, three categories of previous work are closely related to ours: secure data sharing, record linkage, and private record linkage.

Secure data sharing. Private record linkage applies when the underlying data is of sensitive nature. Such privacy aware protocols are categorized under the emergent field of privacy preserving data mining: a topic of wide recent interest. Most work in the area is either in data perturbation, or secure data sharing. Data perturbation algorithms [17, 14] distort individual fields of a database for pri-

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

vacy, but preserve the overall structure of a database. This enables the extraction of aggregate data or patterns without disclosing raw data. On the other hand, secure data sharing considers sharing or querying of selected data while securing all other. Unlike secure data sharing, private record linkage applies when data is heterogeneous.

Many protocols for secure data sharing have been described in the literature. Yao et al [26] describe the first two-party sharing protocol; more protocols and cryptographic constructions that support multiparty sharing are presented in [15]. Secure data sharing protocols assume a *semi-honest* or *honest-but-curious* [3] behavior from the participating parties. This means parties follow the protocol without cheating, but may try to find as much information about one another's databases. Other work by Evfimievski et al [8, 9] investigates the secure mining of association rules, evaluation of breeches, and quantification of privacy.

The work of Agrawal et al [1] defines minimal information sharing in the context of secure data sharing protocols as the set of additional categories of information inferred by parties. The protocols they describe, however, are not useful for matching of heterogeneous data.

Record Linkage. Record linkage is a problem that has received wide attention from different communities. The statistical field is interested in probabilistic approaches to evaluating the similarity of records [24, 10, 13]. In the AI field, training data is fed to distance metrics in order to enhance their ability to recover duplicate records [22]. The information retrieval and database fields are mainly concerned with merging heterogeneous databases and optimizing queries using constructions such as inverted indexes.

To improve the performance of record linkage for large datasets, blocking (sampling) schemes are described in the literature. The essence of blocking is similar to indexing; in the database field, indexing increases the performance of query processing by maintaining tables of related records. Parallel to indexing, several distinctions arise in blocking such as manual vs. automatic, or controlled vs. uncontrolled vocabulary [21]. For generality, it is important that blocking is automated and accommodates any vocabulary.

Blocking schemes reduce the set of candidate record pairs for which more expensive distance metrics, such as TFIDF [21] or Jaro [2], are computed. Baxtor et al [2] analyze several different blocking schemes and compare them for accuracy and reduction ratio. An effective sampling algorithm is described in [12, 11] based on records' textual attributes. Experimentation shows that even a simple blocking scheme can result in significant performance gains, with minimal impact on precision.

Private Record Linkage. Much of the work in private record linkage has been pioneered in the medical field. Of mention is the work of Quantin et al [19, 18], inspired by the confidentiality needs of medical information. In them, Quantin et al describe a keyed hashing transformation protocol for representing epidemiological data as per the requirement of European privacy laws for irreversible transformations of such data. Matching of similar data is computed on the transformed set. More recent trends with privacy needs include cooperation among government agencies, selective sharing of intellectual property, and outsourcing [1]. Churches et al [4] analyze security of their protocols as a function of the number of mediating parties. They refer to their techniques as "blindfolded" record linkage.

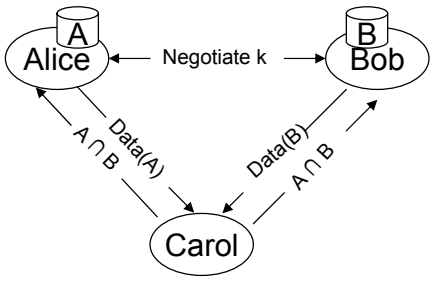


Figure 1: A general third-party matching protocol.

More recent is the work of Ravikumar et al [6] which proposes a secure, stochastic private record linkage protocol that implements all distance metrics where records are representable in a weight vector, such as TFIDF and Soft-TFIDF [5]. Calculation of vectors' similarity relies on a secure intersection algorithm to compare tokens with probability proportional to their weight. Precision asymptotically converges to the true value as the number of record samples increases. Nonetheless, this protocol's use of a secure intersection algorithm translates into expensive computation. Such algorithms rely on commutative hashing based on expensive public key encryptions. Furthermore, the protocol requires multiple occurrences of each record to accurately observe the probabilistic model (i.e., training set). Overall, this work is a valuable contribution that paves the way for two-party private record linkage, however, the constructions upon which it is based lack practical maturity.

3. PRIVATE RECORD LINKAGE

3.1 Overview

Suppose two autonomous parties wish to compute the private record linkage problem on their databases. The task is to design a blocking-compatible private record linkage protocol for enhanced performance. The protocol computes the matching set of records securely, minimizing any information leakage.

We adopt a third party approach to solve this problem. Figure 1 illustrates the communication steps needed. To secure the contents of databases A and B, Alice and Bob must negotiate a secret key and use it to achieve data confidentiality. In this Section, we define our protocol and show how we use *hash signatures* to achieve confidentiality and compactness. Later, we describe several blocking schemes and experimentally verify the performance gains achieved.

3.2 Threat Model and Evaluation

Participants in a private record linkage problem are characterized by *semi-honest*, or *honest-but-curious* behavior [3]. Semi-honest behavior presumes a party will attempt to infer any information possible from the data supplied by other parties. This includes carrying out frequency analysis and known-text attacks of data. We adopt a conservative approach to security based on the premise that an adversary has access to a pool of all known text.

Nonetheless, semi-honest behavior forbids participants from other forms of cheating. Participants do not misrepresent their inputs by supplying false data (spoofing attack) or in-

tentionally hiding parts of their data (hiding attack) [25]. No form of collusion between any party and a third party occurs. Semi-honest behavior is a common requirement for participants in secure data sharing problems in the literature.

In minimal information sharing, security is determined by the additional categories of information divulged in the process of solving a problem. For example, depending on the distance metric used, the third party is given access to information (e.g. weight information) that facilitates matching of records. However, there exist other important categories of information with more tunable levels of exposure. The following categories of information are considered in our analysis of information leakage:

- Database size (DB_{size}).
- Vocabulary size of a database ($Vocab_{size}$).
- Lengths of database records (Rec_{len}).
- Frequency of all tokens in a database (Tok_{frq}), not to be confused with TFIDF's token frequency (TF).

For each category above, we define three levels of exposure:

- Yes (divulged, revealed, determined, exposed): the category is accurately measured by a curious party.
- **inf**: (1) an upper bound is computed on the whole category, or (2) an accurate measure is calculated on a subset of a category.
- No: the category is not divulged, neither is an upper bound divulged.

Determining a level of exposure on the defined categories is a loose characterization of a protocol's privacy. In Section 4, we employ this characterization to analyze the performance of our protocol with respect to blocking scheme alternatives.

3.3 Protocol

Several participants interact in a private record linkage problem. We first define the participants as follows: There are three semi-honest participants: *Alice*, *Bob*, and *Carol*. Alice holds private database A that contains a set of records with a finite number of fields. Bob holds another database B that contains a similar but separate set of records. Alice and Bob are the collaborating parties who wish to find the common records in their databases A and B. Carol is the third party chosen by Alice and Bob to execute the distance metric.

Figure 1 illustrates a general third party matching protocol. The protocol assumes all communication between parties is carried over a secure channel, e.g., via cryptography. A general overview of the steps of our private record linkage protocol that incorporates blocking is as follows:

1. Alice and Bob negotiate a secret key, k, unknown to Carol.
2. Alice and Bob each use a blocking method to generate blocks and assign records of databases A and B to the blocks. Next, Alice and Bob transmit the blocks to Carol.
3. Carol computes a distance on the reduced set of candidate pairs and forwards the results to Alice and Bob.

Third party requirement is common among private record linkage protocols because techniques used in two party protocols fail to correctly represent and match heterogeneous data. Two party protocols rely on double encryptions of data, the output of which are matched. Double encryptions do not preserve properties of heterogeneous data. In [4], the authors further expand on the number of third parties and analyze security as a function of the number of compromised third parties. In [6], the authors describe a third party free protocol for private record linkage, but it achieves poor performance as we mentioned previously.

Third party protocols use keyed transformations of records, e.g keyed hashing, to prevent security breeches due to curious behavior from the third party. Otherwise, the third party can analyze collaborating parties' databases with similar transformations performed on a pool of all known text (i.e brute force).

3.4 Secure Hashing

We introduce the *hash signature* transformation construct as a compact and secure way to represent TFIDF weight vectors. TFIDF is a proven distance metric based on the tokens (words) in a record. Hash signatures use keyed hashing of individual tokens of a record using a pre-negotiated key k. For brevity, k is omitted but implied in the computation of a hash signature. Before presenting further details, it is useful to first introduce TFIDF.

TFIDF. The TFIDF [21] (Token Frequency / Inverse Document Frequency) distance metric is widely used for matching of similar information. TFIDF is based on the intuition that tokens which appear frequently in a given record should be assigned higher weights in that record (TF weight), while tokens which appear frequently in the database as a whole should be generally assigned low weights (IDF weight). For example, consider a database of universities. TFIDF would assign higher weights to tokens such as "PSU" and "Stanford", and lower weights to tokens such as "university" and "college". Records are matched according to a normalization of the linear combination of the TF and IDF weights of similar tokens. Two records A_x and B_y are considered similar if their TFIDF score exceeds some threshold. The TFIDF score is given by:

$$TFIDF(A_x, B_y) = \frac{weight_x \cdot weight_y}{|weight_x| \times |weight_y|} \quad (1)$$

To compute $weight_x$, for each $w_i \in A_x$,

$$weight(x, w_i) = log(TF_{w_i} + 1) \times log(IDF_i) \quad (2)$$

It follows,

$$weight_x = \bigcup_i weight(x, w_i) \quad (3)$$

Alternatively, Eq 1 can be represented as a dot-product of two vectors V_x and V_y [6].

$$V_x = \frac{weight_x}{|weight_x|} \quad (4)$$

Hence, Alice and Bob can independently compute V_x and V_y. The vector lengths are dependent upon the size of the vocabulary set of the two databases.

Database A		Database B	
id	record	id	record
a1	{'a', 'b'}	b1	{'b'}
a2	{'c'}	b2	{'a', 'b'}

Table 1: Databases A and B

	$F[0]$	$F[1]$	$F[2]$	$F[3]$
$HS(a1)$	weight(a1,'b')	0	0	weight(a1,'a')
$HS(a2)$	0	0	weight(a2,'c')	0
$HS(b1)$	weight(b1,'b')	0	0	0
$HS(b2)$	weight(b2,'b')	0	0	weight(b2,'a')

Table 2: Example hash signatures

TFIDF specifies measuring the TF weight of tokens on a record granularity, but measures the IDF weight, i.e. IDF_i, on a common vocabulary of $A \cup B$. For Alice to correctly represent V_x, token information is required from database B, and vice versa. Note that it may be insecure to divulge the IDF weights of the vocabulary of either database. However, it is conceivable to assume that databases A and B are similar, e.g., if database A consists of citation data, database B will also consist of citation data. Hence, separate IDF measurements appear to be good estimates for a common vocabulary.

Hash Signature. As mentioned above, hash signatures are compact and secure representations of TFIDF weight vectors, namely, V_x and V_y. The size of vectors V_x and V_y is not fixed and is based on the data residing in databases A and B. This results in long vectors and requires insecure pre-agreement on vocabulary ordering. A hash signature transformation of A_x denoted as $HS(A_x)$ is a compact, vocabulary-independent representation of the TFIDF weight vector based on a simple hashing function with a small output t, e.g., 10 bits. The hash function is evaluated on each token, and the hash output is an index to a 2^t floating point array where the weight is stored. As with V_x and V_y, the TFIDF score of a pair of records is a dot-product of the hash signatures.

For example, let $A_x = \{w_1, ..., w_n\}$, F be an array of floating point values initialized to 0, and h^t be a hash function with t bit output.

$$\begin{gathered} F[i_1] = weight(x, w_1) \\ F[i_2] = weight(x, w_2) \\ ... \\ F[i_n] = weight(x, w_n) \\ i_l = h^t(w_l), l \in \{1...n\}^1 \end{gathered}$$

$$HS(A_x) = F \tag{5}$$

Example 2. Assume Alice's database A contains 2 records and Bob's database B also contains 2 records as in Table 1. To illustrate how a hash signature is computed, let the hash function h^t output be $t = 2$. As mentioned previously, the hash function performs keyed hashing with an omitted key k: $h^t('a') = 3$, $h^t('b') = 0$, and $h^t('c') = 2$. Each hash signature is composed of a floating point array of weights, as shown in Table 2. The array position where a weight of a particular token is stored depends on the hash output of the token. For example, weight(a1,'b') is stored in the entry with index 0, because $h^t('b') = 0$. □

[1] *Actually*, $h^t(w_i||k)$ is computed, but k is omitted for brevity. The symbol $'||'$ denotes string concatenation.

3.5 Analysis

Notice that since the hash output size, t, is relatively small, there is a high probability of collisions. This means distinct tokens of a record may map to the same hash signature entry. Likewise, dissimilar tokens of a record pair may map to the same entry. This can result in inaccurate TFIDF scoring. In our implementation, if multiple tokens of a record map to the same entry, only the smallest weight is stored to minimize the effect on TFIDF scoring. Assuming that every collision incident results in incorrect matching:

Lemma 1. *The worst case performance of our protocol using hash signatures is* $\frac{2^t!}{(2^t-n)!\times 2^{nt}}$*, where* n *is the number of different tokens in* $A_x \cup B_y$*.* ■

Proof. Assume a worst case performance such that every collision occurrence results in a mismatch. We want to show the worst case performance of TFIDF using hash signatures is $\frac{2^t!}{(2^t-n)!\times 2^{nt}}$ of the performance of TFIDF. But, given $HS(A_x)$, and $HS(B_y)$, the number of permutations such that two different tokens in the the pair map to the same array entry $= \begin{pmatrix} 2^t \\ n \end{pmatrix} \times n!$. By dividing the total number of possibilities, we have $\frac{\begin{pmatrix} 2^t \\ n \end{pmatrix} \times n!}{2^{nt}} = \frac{2^t!}{(2^t-n)!\times 2^{nt}}$. (q.e.d)

Hence, for $n \ll 2^t$, the worst case achieves performance above 95% of performance of TFIDF. Note, t is a design parameter that can be adjusted to satisfy $n \ll 2^t$.

A further modification geared at conserving disk space is possible with hash signatures. Since $n \ll 2^t$, hash signatures are sparse weight arrays. It follows that the set of database records is an instance of a sparse matrix. There are many ways to compactly represent sparse matrices, including smaller arrays of positions and corresponding values or a linked list of position/value pairs. However, we consider a technique known as *run-length encoding*, which lists each weight followed by the number of zeros suppressed. Run-length encoding is suitable to our scheme because it is easy to implement and features a small overhead, yet has been found to achieve up to 70% compression [21].

4. BLOCKING SCHEMES

Among many variations, in our implementation, we use token blocking [5] – every pair of records becomes a candidate pair if they share at least one token. A separate block is associated with every token, containing records in which the token appears.

Phase 2 Blocking. The structure of our hash signatures enables augmenting the blocking process with a second phase executed on records of the same block to further reduce the set of candidate pairs. We find that the well-known Jaccard [5] metric is readily compatible. Jaccard is computed on a binary representation of a record's hash signature, such that non-zero weights are treated as a binary 1

and zero weights as a binary 0. Let D_A and D_B be the binary representations of A_x and B_y, respectively, the Jaccard metric is given by,

$$Jaccard(D_A, D_B) = \frac{|D_A \cap D_B|}{|D_A \cup D_B|} \quad (6)$$

The use of Jaccard for blocking appears to further increase performance, because it is computed very efficiently. The details are investigated in our experiments.

Information Leakage. The level of information leakage is evaluated for our protocol with respect to each blocking scheme introduced. It is important to note, however, there are subtle differences between the levels of sharing defined. To better illustrate this, consider a database of records mapped to hash signatures. It is easy to see that DB_{size} is divulged as it equals the number of hash signature entries. However, only **inf** Rec_{len} is divulged because a hash signature may incur collisions. Furthermore, **inf** $Vocab_{size}$ is revealed because the number of different entries in the set of hash signatures necessarily means different tokens. However, Tok_{frq} is not revealed because collisions do not preserve an accurate measure of the frequency of tokens.

On the other hand, if a subset of the hash signatures is available, only **inf** DB_{size} is divulged. However, Rec_{len} is not divulged, because it is an inaccurate measure (due to collisions) that is revealed on a subset of all records. Lastly, $Vocab_{size}$ is also not revealed for the same reason.

Assumptions. We partially base our analysis of security on several assumptions. Records are non-empty and do not solely consist of *stop list tokens* – very commonly occurring tokens in a database. We further assume that stop list tokens are not interesting: a measure of a category is considered accurate, even if it fails to represent stop list tokens.

In addition, if two hash signatures possess a similar set of weights stored, this does not necessarily imply that the hash signatures are the same. They are only considered the same if the weights reside in the same array entries. The reason for this will become more apparent in the following subsections.

4.1 Baseline Approach

Assume two parties Alice and Bob wish to compute the matching problem on their databases A and B, respectively. A is a set of records $A_1, ..., A_n$ and B is a set of records $B_1, ..., B_m$. Alice and Bob share a secret key k that is tacitly used in the computation of hash signatures. Carol is the mediating third party. In the "baseline" scheme, the matching problem is computed without the use of any blocking. The interest of this scheme is purely experimental. Alice and Bob transmit sequences of the hash signatures of databases A and B to Carol. Carol computes the matching problem on all pairs of A and B, and identifies matching records to Alice and Bob. This scheme proceeds as follows:

1. For each $A_x \in A$, Alice computes $HS(A_x)$. Likewise, for each $B_y \in B$, Bob computes $HS(B_y)$.

2. Let A_{hs} denote all $HS(A_x)$ that Alice transmits to Carol. Let B_{hs} denote all $HS(B_y)$ which Bob transmits to Carol.

3. For each $D_A \in A_{hs}$, D_A is paired with every $D_B \in B_{hs}$ for computing of the distance metric.

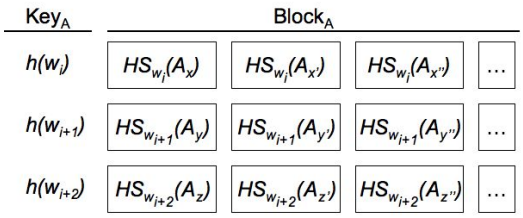


Figure 2: Phase 1 blocking in Simple blocking

This scheme does not employ any blocking, resulting in $O(nm)$ load. The additional categories of information revealed to Carol, I_0, in this scheme include: DB_{size}, **inf** $Vocab_{size}$, and **inf** Rec_{len} of databases A and B, because of the possibility of collisions in hash signatures. Tok_{frq} is not revealed.

4.2 Simple Blocking

In the Simple blocking scheme, token blocking is used to enhance the performance of the baseline scheme. The collaborating parties arrange records in blocks and transmit the blocks to Carol. The protocol steps are as follows:

1. For each $w_i \in A_x$, Alice computes $HS_{w_i}(A_x)$ and stores it in R_{w_i}, the block of w_i. Similarly, for each $w_j \in B_y$, Bob computes $HS_{w_j}(B_y)$ and stores it in S_{w_j}, Bob's block of w_j.

2. Let $Key_A \rightarrow Block_A$ denote all mappings $h(w_i) \rightarrow R_{w_i}$ that Alice transmits to Carol. Let $Key_B \rightarrow Block_B$ denote all mappings $h(w_j) \rightarrow S_{w_j}$, which Bob transmits to Carol.

3. For each $w \in Key_A$, if $w \in Key_B$, compute the Jaccard metric on every pair $D_A \in Block_A$ and $D_B \in Block_B$.

4. If $Jaccard(D_A, D_B) > threshold$, D_A and D_B are paired for computing of the distance metric.

In step 2 of this scheme, h is a keyed hashing function that provides confidentiality to the block identifier, i.e. the token; the key is concatenated analogous to hash signatures. Any secure hashing algorithm may be used. In our case, we implement SHA-1.

I_1, the additional categories of information divulged to Carol in Simple blocking include: $Vocab_{size}$ because token blocking generates a separate block for each token in a record, and **inf** DB_{size} of databases A and B. Further, since Carol has no way of knowing whether hash signatures appearing in different blocks with permuted weights represent the same original record, only **inf** DB_{size} is determined by Carol. For the same reason and because of hash signature collisions, only **inf** Rec_{len} is divulged.

Finally, Tok_{frq} of databases A and B is divulged to Carol. Each block contains a list of hash signatures where the token appears, thereby an accurate measure of frequency of a token is revealed. This may prove to be a vulnerability to statistical attacks, such as Zipf [21] distribution. A technique to thwart such attacks is considered in Section 6.

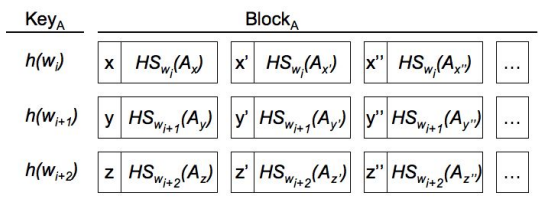


Figure 3: Phase 1 blocking in Record-aware blocking

4.3 Record-aware Blocking

In Simple blocking, the assignment of hash signatures to blocks equivocates a record across blocks where it is placed. The reason is that in step 1 of Simple blocking, the hash signature is computed with respect to the block (HS_{w_i}). Hence, the same record is indistinguishable to the third party when placed in multiple blocks. This means Simple blocking will compute duplicate distance metrics on the same pair of records proportional to the number of common blocks in which they appear. A relatively high ratio of common blocks per record pair can render the blocking scheme extremely inefficient. Record-aware resolves this by coupling an id with the hash signature of each record. The ids of every pairing made are maintained to ensure uniqueness of a pair. The blocking steps of Record-aware are as follows:

1. For each $w_i \in A_x$, Alice computes $HS_{w_i}(A_x)$ and stores the pair $[x, HS_{w_i}(A_x)]$ in R_{w_i}, the block of w_i. Similarly, for each $w_j \in B_y$, Bob computes $HS_{w_j}(B_y)$ and stores the pair $[y, HS_{w_j}(B_y)]$ in S_{w_j}, Bob's block of w_j.

2. Let $Key_A \rightarrow [ID_A, Block_A]$ denote all mappings $h(w_i) \rightarrow R_{w_i}$ that Alice transmits to Carol. Let $Key_B \rightarrow [ID_B, Block_B]$ denote all mappings $h(w_j) \rightarrow S_{w_j}$, which Bob transmits to Carol.

3. For each $w \in Key_A$, if $w \in Key_B$, Carol computes the Jaccard metric on every pair $D_A \in Block_A$ and $D_B \in Block_B$, such that $(ID_A, ID_B) \notin Paired$. $Paired$ is used to ensure that duplicate distance metrics on the same pair are not computed.

4. If $(ID_{D_A}, ID_{D_B}) \notin Paired$ and $Jaccard(D_A, D_B) > threshold$, D_A and D_B are paired for computing of the distance metric. Update $Paired = Paired \cup \{(ID_{D_A}, ID_{D_B})\}$

The increase in efficiency in this scheme is attained at the expense of additional data leakage in comparison to Simple blocking. This scheme divulges all designated categories of information to Carol. Unlike Simple blocking, Carol determines DB_{size} of databases A and B, because an id is coupled with each hash signature that uniquely identifies the record which it represents. Rec_{len} are also divulged for the same reasons. As in Simple blocking, $Vocab_{size}$ and Tok_{frq} are revealed. This scheme provides the greatest exposure in terms of information leaked to Carol.

4.4 Frugal Third Party Blocking

In the previous two schemes, Alice and Bob transmit all of their blocks to Carol, even if only a fraction of the blocks are similar. Alternatively, Alice and Bob may find what blocks they have in common and only transmit those blocks. This arrangement has the advantage of minimizing communication size between participants. Further, the amount of data storage maintained by Carol for the matching problem is reduced, and Carol is relieved from the task of finding intersection between the blocks, resulting in less blocking and matching time. Moreover, from a monetary cost standpoint, any computation performed by Carol is associated with a much greater price than computations performed by Alice or Bob. Carol is merely a mediating party which may charge a significant fee for facilitating matching between collaborating parties. It follows that it may be more cost-effective for Alice and Bob to perform as much computation as possible, even if such computation is orders of magnitudes greater than a logically equivalent computation performed by Carol.

Scheme	Information Categories			
	DB_{size}	$Vocab_{size}$	Rec_{len}	tok_{frq}
Baseline	Yes	**inf**	**inf**	No
Simple	**inf**	Yes	**inf**	Yes
Record-aware	Yes	Yes	Yes	Yes
Frugal Third Party	**inf**	**inf**	No	**inf**

Table 3: Summary of information leakage

Frugal Third Party blocking employs a secure intersection algorithm [23, 6] to compute the intersection set size using commutative one-way hash functions. In our implementations, we use RSA private keys with a common modulus. The scheme steps are as follows:

1. For each $w_i \in A_x$, Alice computes $HS_{w_i}(A_x)$ and stores the pair $[x, HS_{w_i}(A_x)]$ in R_{w_i}, the block of w_i. Similarly, for each $w_j \in B_y$, Bob computes $HS_{w_j}(B_y)$ and stores the pair $[y, HS_{w_j}(B_y)]$ in S_{w_j}, Bob's block of w_j.

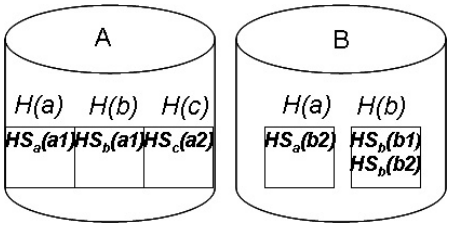
2. Let $Key_A \rightarrow [ID_A, Block_A]$ denote all mappings $h(w_i) \rightarrow R_{w_i}$. Let $Key_B \rightarrow [ID_B, Block_B]$ denote all mappings $h(w_j) \rightarrow S_{w_j}$. Using the secure intersection algorithm, find $Key_A \cap Key_B$. Alice transmits to Carol all $w \rightarrow [ID_A, Block_A]$ and Bob transmit all $w \rightarrow [ID_B, Block_B]$, such that $w \in Key_A \cap Key_B$.

3. For each $w \in Key_A \cap Key_B$, Carol computes the Jaccard metric on every pair $D_A \in Block_A$ and $D_B \in Block_B$, such that $(ID_A, ID_B) \notin Paired$. As in Record-aware, $Paired$ is used to ensure that duplicate distance metrics on the same pair are not computed

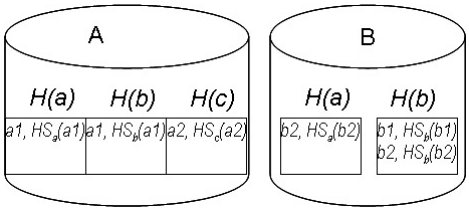
4. Finally, if $(ID_{D_A}, ID_{D_B}) \notin Paired$ and $Jaccard(D_A, D_B) > threshold$, D_A and D_B are paired for computing of the distance metric. Update $Paired = Paired \cup \{(ID_{D_A}, ID_{D_B})\}$

In this scheme, some additional categories of information are revealed only to Alice and Bob, while others only to Carol. Let I_{3AB} denote the additional categories revealed to Alice and Bob and I_{3C} denote categories revealed to Carol.

Alice and Bob predetermine the blocks which are shared by them. Hence, I_{3AB} contains $Vocab_{size}$, meaning Alice learns the vocabulary size of database B and Bob learns the vocabulary size of Alice. Since Carol does not receive all blocks, I_{3C} includes **inf** DB_{size} and **inf** $Vocab_{size}$ because an accurate measure is determined on a subset of the categories. Further, Carol only determines an accurate Tok_{frq}



(a) Blocks of Simple blocking



(b) Blocks of Record-aware and Frugal Third Party

Figure 4: Blocking Example

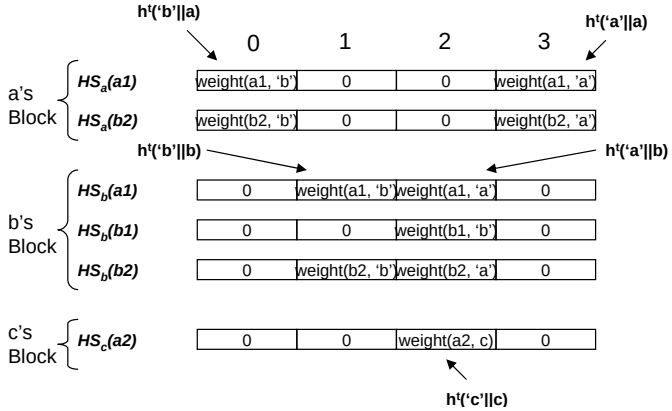


Figure 5: Hash signature contents

of a subset of all tokens because she only receives a subset of all blocks, hence **inf** Tok_{frq}. However, Carol does not determine Rec_{len} because an inaccurate measure is possible on merely a subset of records. The security of this scheme introduces a level of uncertainty to the categories divulged to Carol at the cost of increased computational cost and exposure of some categories to Alice and Bob. From an information leakage standpoint, Frugal Third Party is more useful when security with respect to the third party is of prime concern.

Example 3. Here, we present an example that illustrates the working of Simple, Record-aware, and Frugal Third Party blocking schemes. We consider databases A and B as defined in Table 1.

1. **Step 1.** Figure 4(a) illustrates how a record's hash signature is arranged in blocks that correspond to the tokens contained in the record. For example, record a1 contains tokens 'a' and 'b'; hence, it appears in the block corresponding to a as HS_a and the block corresponding to b as HS_b. Notice that a hash signature representation appears differently depending on the block where it is stored. This is accomplished by a concatenation of the block's token to the hash function used to generate the hash signature (note a key is also concatenated).

   The blocks of Record-aware and Frugal Third Party blocking schemes appear a little differently, as illustrated in Figure 4(b). As mentioned above, an id is coupled with each hash signature, so identical pairs are not matched for similarity multiple times. Assuming t is 2-bits, and $h^t('a'||a) = 3$, $h^t('b'||a) = 0$, $h^t('c'||c) = 2$, $h^t('a'||b) = 2$, $h^t('b'||b) = 1$, The hash signatures are computed as shown in Figure 5.

2. **Step 2.** In Simple and Record-aware blocking, Alice and Bob readily transmit their blocks of hash signatures computed in the last step to Carol. Alice transmits 3 blocks to Carol, while Bob transmits 2 blocks to Carol. However, in Frugal Third Party, Alice and Bob first determine the matching block names using a secure intersection algorithm. It is determined that only two blocks match, corresponding to a and b. Hence, Alice and Bob both transmit 2 blocks to Carol.

3. **Step 3.** Carol computes the distance metric on hash signatures which appear in common blocks. In Simple blocking, all records of the parties which appear in similar blocks are considered. This results in the following pairings: ($HS_a(a1)$, $HS_a(b2)$), ($HS_b(a1)$, $HS_b(b2)$), and ($HS_b(a1)$, $HS_b(b1)$). On the other hand, Record-aware and Frugal Third Party only pair a1 and b2 once, instead of twice. Hence, only two pairings are made: ($HS_a(a1)$, $HS_a(b2)$) and ($HS_b(a1)$, $HS_b(b1)$).

4. **Step 4.** The Jaccard metric is computed on a binary equivalent of each hash signature. Figure 6 illustrates binary equivalents of the hash signatures. It follows:

   - $Jaccard(HS_a(a1), HS_a(b2)) = 1$
   - $Jaccard(HS_b(a1), HS_b(b2)) = 1$
   - $Jaccard(HS_b(a1), HS_b(b1)) = .5$

   In the absence of blocking, 4 distance metrics are calculated. If Jaccard $threshold > .5$, then 2 distance metric are calculated in Simple blocking, resulting in 50% reduction ratio. Only 1 metric is calculated in the Record-aware and Frugal Third Party schemes resulting in a reduction ratio of 25%. □

	0	1	2	3
$HS_a(a1)$	1	0	0	1
$HS_a(b2)$	1	0	0	1
$HS_b(a1)$	0	1	1	0
$HS_b(b1)$	0	0	1	0
$HS_b(b2)$	0	1	1	0
$HS_c(a2)$	0	0	1	0

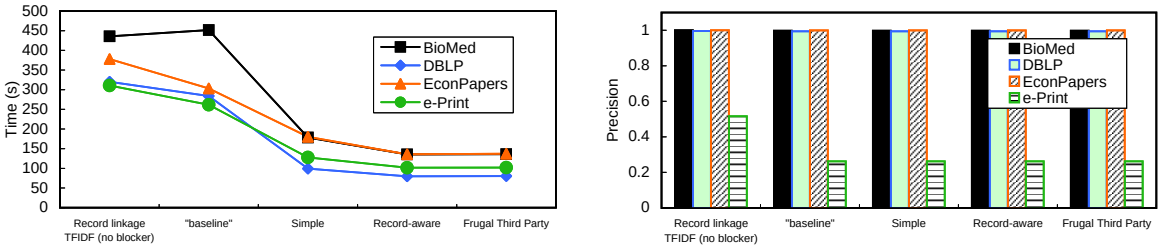
Figure 6: Binary representations of hash signatures

5. EXPERIMENTAL VALIDATION

We have implemented our schemes based on the SecondString package [5], and have tested it using a range of datasets

Dataset	Domain	A size	B size	A ∩ B size	# of tokens per record (avg)
BioMed	Medicine	5000	5000	2000	25.44
DBLP	CompSci	5000	5000	2000	14.68
EconPapers	Economics	5000	5000	4000	21.23
e-Print	Physics	5000	5000	2000	13.83

Table 4: Experimental Datasets 1



(a) Total time (blocking + matching)

(b) Precision

Figure 7: Comparison of blocking + matching time of four implementations

.

consisting of citation data. Each citation is parsed as a set of fields and only a random permuted subset of the fields are stored to achieve a level of heterogeneity. The choice of citation data was influenced by Ravikumar et al [6]. As an evaluation metric, *precision* is defined as the fraction of pairs matched by the distance metric that are correct, and *reduction ratio* as the ratio between candidate pair count from blocking and pair count from a pair-wise comparison.

5.1 Blocking

Figure 7(a) proves our claims of increased performance due to blocking, by comparing the blocking schemes against the baseline scheme and record linkage TFIDF that doesn't use blocking. Note, computation time due to the secure intersection algorithm in the Frugal Third Party scheme has been factored out. Table 4 lists the datasets used in this experiment and their characteristics. For all cases, the blocking techniques require a fraction of the time to solve the same problem, often at a negligible cost of precision. Figure 7(b) illustrates the precision observed for each dataset. The precision observed for the e-Print dataset deviates from other datasets. However, even in the non-private protocol, low precision is observed. Nonetheless, this is magnified in the private record linkage protocols.

Moreover, it is observed that representing records as hash signatures increases performance. We attribute this to the way hash signatures are represented, which is more compact than weight vectors in record linkage.

5.2 Blocking Schemes

To verify the properties of the proposed blocking scheme, several experiments were conducted using DBLP citations as in Table 5.

In Figure 8(a), the total time expended by the third party is measured as a function of dataset used. Simple blocking is the least efficient for the reason that every pair of records may be measured for similarity more than once. The total time expended for Simple and Record-aware blocking does not follow any specified pattern because of additional factors that come into play, such as the total number of blocks in each database, or the number of shared blocks. On the other hand, in Frugal Third Party, as the number of shared records increases, total third party time also increases because only common blocks are transmitted over to the third party. Clearly, the number of common blocks is likely to increase relative to the number of shared blocks. When A and B have the same records, no third party gains are observed by Frugal Third Party over Record-aware, resulting in transmission of almost all blocks, congruent to Simple and Record-aware.

Dataset	A size	B size	A ∩ B size
1	2500	2500	0
2	2500	2500	100
3	2500	2500	250
4	2500	2500	500
5	2500	2500	1000
6	2500	2500	2500

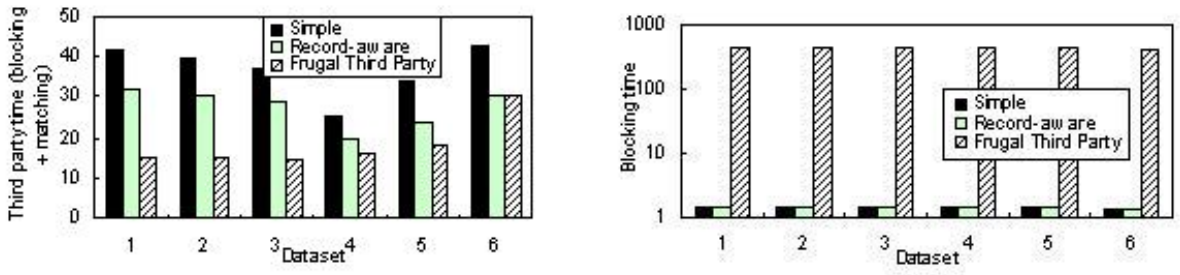
Table 5: Experimental Datasets 2

Frugal Third Party reduction of third party time and restriction on privacy divulged to third party comes at an overwhelming cost of collaborating party blocking. Figure 8(b) illustrates the average blocking time of Frugal Third Party, Simple, and Record-aware schemes. It eludes to 4 orders of magnitude difference in blocking time.

The reduction ratio observed by two phase blocking is dependent on characteristics of the dataset, such as $Vocab_{size}$ and Tok_{frq}. Nonetheless, reduction ratio is directly responsible for the matching time needed to solve the problem. Figure 9 illustrates the linear relationship between reduction ratio and matching time.

5.3 Two-phase Blocking

The second phase of blocking, Jaccard metric, depends on basic set operations to further reduce candidate pairs that undergo TFIDF scoring. The *threshold* used for Jaccard



(a) Total third party time

(b) Blocking time of collaborating parties

Figure 8: Comparison of three blocking alternatives

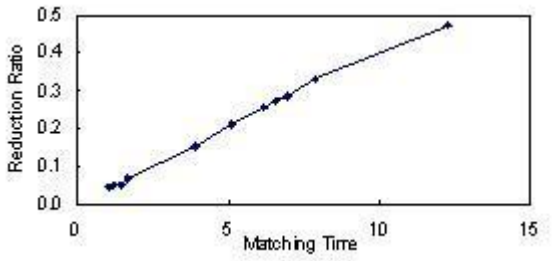


Figure 9: Relationship between reduction ratio and matching time

blocking is tunable for enhanced control over the level of blocking achieved. In Figure 7(a), the gains resultant from predominantly phase 1 blocking are observed. In practice, using phase 2 Jaccard, we are able to control the gains by adjusting the *threshold*. Figure 10 illustrates the performance increase when the Jaccard *threshold* is increased to .25. Experiments of the previous subsection all used *threshold* = .1.

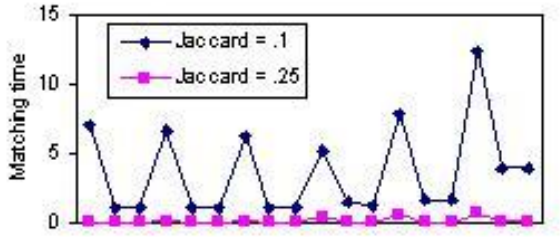


Figure 10: Relationship between reduction ratio and matching time for Dataset 1-6

6. OTHER ISSUES

The privacy of our protocol can be further increased by using techniques such as *chaffing* and *winnowing* [20]. Chaffing refers to the act of augmenting data with additional dummy data, indiscernible from other data, i.e. noise. Winnowing is the reverse process that eliminates dummy from real data. Chaffing is performed by the collaborating parties prior to transmitting data to the third party. After the results are computed and returned to the collaborating parties, all dummy data is winnowed appropriately.

Chaffing and winnowing techniques address the problem introduced by a Zipf [21] analysis of data blocks that we identified earlier. The problem is solved by fixing the content size of each block to make them appear indifferent, through the addition of noise. Nevertheless, this comes at the expense of increased cost and memory requirements. Our current implementations do not consider chaffing and winnowing.

7. CONCLUSION

In this paper, we described a secure protocol for record linkage and several schemes that achieve secure blocking. In the protocol, the collaborating parties use TFIDF to independently calculate and generate weight vectors, represented as hash signatures. This protocol requires a key to be negotiated by the collaborating parties, unknown to the third party.

The blocking schemes described are characterized by varying privacy as per minimum information sharing at the expense of performance. We define a loose characterization of information leakage to analyze the privacy of our protocols as a function of blocking scheme. Simple blocking arranges hash signatures in blocks, but a pair may be computed for similarity more than once if they are located in more than one common block. Record-aware solves this issue by coupling an identifier with every hash signature. Frugal Third Party provides the highest level of privacy with respect to information divulged to the third party, but requires the use of public-key dependent secure intersection algorithm, resulting in heavy computational costs on the collaborating parties. The performance of our blocking schemes was further enhanced by introducing a second phase of blocking: Jaccard.

Overall, parallel to record linkage, the use of blocking in private record linkage to enhance performance is verified using analytic and empirical validations. In some cases, blocking increases the security en lieu of performance of the protocol.

Future Work. It is interesting to investigate how other distance metrics and blocking schemes can be ported to the secure matching arena. In [6], the authors address other distance metrics and how they may be ported into secure record linkage. Clearly, similar techniques as described in this paper may also apply to those metrics.

8. REFERENCES

[1] Rakesh Agrawal, Alexandre Evfimievski, and Ramakrishnan Srikant. Information sharing across private databases. In *Proceedings of ACM SIGMOD*, pages 86–97, 2003.

[2] Rohan Baxter, Peter Christen, and Tim Churches. A comparison of fast blocking methods for record linkage. In *Proceedings of 9th ACM SIGKDD Workshop on Data Cleaning, Record Linkage and Object Consolidation*, 2003.

[3] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *STOC 96*, pages 639–648, 1996.

[4] Tim Churches and Peter Christen. Some methods for blindfolded record linkage. *BMC Medical Informatics and Decision Making*, 4(9), 2004.

[5] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for matching names and records. In *KDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[6] William Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A secure protocol for computing string distance metrics. In *Proceedings of ICDM Workshop on Privacy and Security Aspects of Data Mining*, 2004.

[7] W. Du and M. Atallah. Potocols for secure remote database access with approximate matching. In *1st Workshop on Security and Privacy in E-Commerce*, 2000.

[8] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of SIGMOD*, 2003.

[9] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In *Proceedings of 8th ACM SIGKDD*, pages 217–228, 2002.

[10] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[11] Luis Gravano, Panagiotis G. Ipeirotis, Jagadish Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava. Approximate string joins in a database (almost) for free. In *Proceedings of 27th VLDB*, pages 491–500, 2001.

[12] Luis Gravano, Panagiotis G. Ipeirotis, Koudas Koudas, and Divesh Srivastava. Text joins in an rdbms for web data integration. In *Proceedings of 12th WWW*, January 01 2003.

[13] Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. In *Proceedings of ACM SIGMOD*, pages 127–138, 1995.

[14] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. Random data perturbation techniques and privacy preserving data mining. In *Proceedings of ICDM*, pages 160–164, 2003.

[15] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay — a secure two-party computation system. In *Proceedings of 11th USENIX Security Symposium*, August 2004.

[16] H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[17] Huseyin Polat and Wenliang Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proceedings of ICDM*, 2003.

[18] Catherine Quantin, H. Bouzelat, F. Allaert, A. Benhamiche, J. Faivre, and L. Dusserre. How to ensure data security of an epidemiological follow-up: quality assessment of an anonymous record linkage procedure. *International Journal of Medical Informatics*, 49(1):117–122, 1998.

[19] Catherine Quantin, H. Bouzelat, and L. Dusserre. A computerized record hash coding and linkage procedure to warrant epidemiological follow-up data security. *Studies in Health Technology and Informatics*, 43:339–342, 1997.

[20] Ronald Rivest. Chaffing and winnowing: Confidentiality without encryption. *MIT, Internal Paper*, 1998.

[21] Gerard Salton, editor. *Automatic Text Processing*. Addison Welsley, 1989.

[22] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.

[23] Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to associate rule mining. *Journal of Computer Security*, 2004. To Appear.

[24] W. E. Winkler. Matching and record linkage. *Business Survey Methods*, pages 355–384, 1995.

[25] L. Xiong, S. Chitti, and L. Liu. Topk queries across multiple private databases. In *25th ICDCS. To appear*, 2005.

[26] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Symposium on FOCS*, pages 160–164, 1982.

Effective and Scalable Solutions for Mixed and Split Citation Problems in Digital Libraries

Dongwon Lee*
Penn State / USA
dongwon@psu.edu

Byung-Won On
Penn State / USA
on@cse.psu.edu

Jaewoo Kang
NCSU / USA
kang@csc.ncsu.edu

Sanghyun Park
Yonsei Univ. / Korea
sanghyun@cs.yonsei.ac.kr

ABSTRACT

In this paper, we consider two important problems that commonly occur in bibliographic digital libraries, which seriously degrade their data qualities: *Mixed Citation (MC)* problem (i.e., citations of different scholars with their names being homonyms are mixed together) and *Split Citation (SC)* problem (i.e., citations of the same author appear under different name variants). In particular, we investigate an effective yet scalable solution since citations in such digital libraries tend to be large-scale. After formally defining the problems and accompanying challenges, we present an effective solution that is based on the state-of-the-art sampling-based approximate join algorithm. Our claim is verified through preliminary experimental results.

1. INTRODUCTION

Bibliographic Digital Libraries (DLs), such as DBLP, CiteSeer or e-Print arXiv, contain a large number of citation[1] records. Such DLs have been an important resource for academic communities since scholars often try to search for relevant works from DLs. Researchers also use the citation records in order to measure the publication's impact in the research community. In addition, citations are often used when users search for articles of interest. Therefore, it is important to keep the citations of DLs consistent and up-to-date. However, due to data entry errors, imperfect citation gathering software or common author names, DLs often have many errors in their citation collections. In particular, we focus on two problems that commonly occur in many existing DLs as follows.

First, when two scholars have the same name spellings, their citation data are mistakenly merged into a single collection, leading to an incorrect citation analysis results. We call this as **Mixed Citation (MC)** problem. For instance,


Dongwon Lee

List of publications from the DBLP Bibliography Server - FAQ

Coauthor Index - Ask others: ACM DL - ACM Guide - CiteSeer - CSB - Google

2004

27 Alberto H. F. Laender, Dongwon Lee, Marc Ronthaler: Sixth ACM CIKM International Workshop on Web Information and Data Management (WIDM 2004), Washington, DC, USA, November 12-13, 2004. ACM 2004

26 EE Bo Luo, Dongwon Lee, Wang-Chien Lee, Peng Liu: QFilter: fine-grained run-time XML access control via NFA-based query rewriting. CIKM 2004: 543-552

25 EE Dongwon Lee, Divesh Srivastava: Counting Relaxed Twig Matches in a Tree. DASFAA 2004: 88-99

24 EE Yoojin Hong, Byung-Won On, Dongwon Lee: System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach. ECDL 2004: 134-144

23 EE Robert J. Kauffman, Dongwon Lee: Should We Expect Less Price Rigidity in the Digital Economy? HICSS 2004

22 EE Byung-Won On, Dongwon Lee: PaSE: Locating Online Copy of Scientific Documents Effectively. ICADL 2004: 408-418

21 Robert J. Kauffman, Dongwon Lee: Price Rigidity on the Internet: New Evidence from the Online Bookselling Industry. ICIS 2004: 843-848

Figure 1: Mixed citations of "Dongwon Lee" in DBLP. Boxed ones are by another "Dongwon Lee."

Figure 1 illustrates a collection of citation data by one of the authors, "Dongwon Lee", in DBLP. Note that two citations by "another" scholar with the same name spelling are listed (boxed ones). The reason of this mixture is that there exist two scholars with the name "Dongwon Lee" – a computer scientist at Penn State and an MIS scholar at U. Minnesota – with somewhat overlapping domains of interests.

Second, due to various reasons, DLs tend to keep the citations of single author under various *name variants*[2]. We call this as **Split Citation (SC)** problem. For instance, imagine a scholar "John Doe" has published 100 articles. However, a DL keeps two separate name variants, "John Doe" and "J. D. Doe", each of which contains 50 citations. In such a case, users searching for all the articles of "John Doe" will get only 50 of them. Similarly, any bibliometrical study would underestimate the impact of the author "John Doe", splitting his fare share into "John Doe" and "J. D. Doe" incorrectly. Such a problem of ambiguous author names exists in many of existing DLs, as illustrated in Figure 2, where a renowned computer scientist "Jeffrey D. Ullman" appear under 10 different name variants in ACM Portal's DL.

In essence, both MC and SC problems cannot be completely avoided unless each person carries a universal ID. Note that these problems would still occur even if digital

*The author is partially supported by IBM Eclipse Innovation Award (2004) and Microsoft SciData Award (2005).

[1]A *citation* or *reference* is a bibliographic entity that consists of various fields (e.g., author, title, venue or year).

[2]In this paper, the *name variants* refer to the different spellings of author names that are in fact referring to the same person.

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

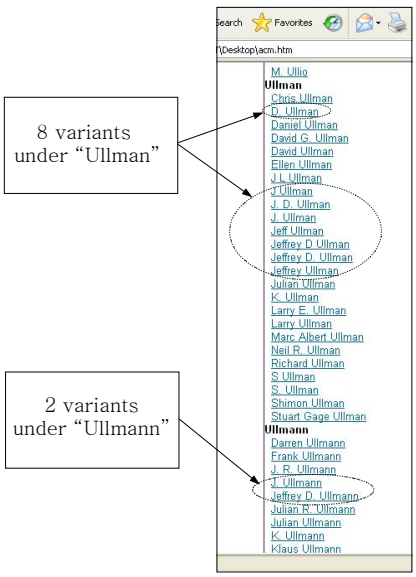


Figure 2: Split citations of "Jeffrey D. Ullman" in ACM Portal.

object identifier (DOI)[3] system is fully adopted, since it usually does not govern the identity of a person or her name. In this paper, therefore, we investigate efficient solutions for these problems.

2. BACKGROUND

Related Work. In [14], we investigated issues related to system support for both problems, and in [20], we explored the split citation problem. Han et al. [12] proposed two supervised learning-based approaches for a related problem. Their problem can be viewed as a kind of the SC problem in our jargon although they do not explicitly define the problem. Furthermore, their approach is not scalable to handle large-scale DLs. On the other hand, we investigate both the MC and SC problems, and present scalable solutions. Nevertheless, we also tested their ideas of supervised learning methods in the step 2 of the name disambiguation algorithm (Section 4.2). The goal of our study is not to compare supervised methods against unsupervised ones, but to explore the combinations of alternatives that give good scalability/accuracy trade-offs in the two-step approaches. We recently learned that [2] introduces a clustering-based solution to a problem similar to our MC problem, and the performance comparison is currently underway. ALIAS system in [22] proposes a framework to detect duplicate entities such as citations or addresses, but its focus is on the learning aspect.

Since the core technique of our proposals in this paper is to match two related citations (i.e., citation matching), our work is closely related to more general class of problems, known as various names – record linkage (e.g., [10, 3]), citation matching (e.g., [19]), identity uncertainty (e.g., [21]), merge-purge (e.g., [13]), object matching (e.g., [5]), duplicate detection (e.g., [22, 1]), approximate string join (e.g., [11]) etc.

String similarity measures used in our work were proposed by Jaro [16] and Winkler [25]. Bilenko et al. have studied name matching for information integration [3] using string-based and token-based methods. Cohen et al. have also compared the efficacy of string-distance metrics, like JaroWinkler, for the name matching task [6]. In DLs, this problem is called citation matching. In the citation matching domain, [18] experimented with various distance-based algorithms with a conclusion that word based matching performs well. We have implemented all these methods in the second step of your algorithm and compared their efficacy to other methods.

Before we can process citations, we assume that field segmentation and identification has been completed using some methods like one in [4]. Blocking was first proposed by Kelley et al. [17] in the record linkage literature. Our sampling idea can be viewed as a blocking scheme and is also similar in flavor to the two-step citation matching schemes proposed in [15, 19] where initial rough but fast clustering (or "Canopy") is followed by more exhaustive citation matching step.

Another stream of works that are relevant to our work is name/entity disambiguation and authority controls in NLP community. For instance, works done in [24] aim at detecting name variants automatically using data mining or heuristics techniques, but do not consider the issue of scalability nor in the context of digital libraries. Similarly, [9] introduces a method to find matching variants of named entity in a given text such as project name (e.g., DBLP vs. Data Base and Logic Programming). [23] discusses an effort to standardize author names using a unique number, called INSAN, and [8] is a recent implementation for name authority control, called HoPEc. On the contrary, we focus more on two specific problems relevant to citations of digital libraries.

Preliminaries. In this Section, we introduce a technique that our solutions exploit. One of the state-of-the-art sampling techniques that satisfy both criteria (i.e., being fast and accurate) is the *sampling-based join approximation* method recently proposed by [11]. We adopt it to our context as follows: Their main idea is that if, for each string n_i, one is able to extract a small sample S that contains mostly strings suspected to be highly similar to n_i, then this sample S serves as a candidate set, and the remaining strings can be quickly ignored (i.e., pre-filtering). To get the "good" sample S, imagine each token from all strings has an associated weight using the TFIDF metric in IR (i.e., common tokens in strings have lower weights while rare ones have higher weights). Then, each string t is associated with its token weight vector v_t. Suppose that, for each string t_q in a string set R_1, we want to draw a sample of size S from another string set R_2 such that the frequency C_i of string $t_i \in R_2$ can be used to approximate $sim(v_{t_q}, v_{t_i}) = \sigma_i$. That is, σ_i can be approximated by $\frac{C_i}{S} T_V(t_q)$, where $T_V(t_q) = \sum_{i=1}^{|R_2|} \sigma_i$. Then, put t_i into a candidate set only if $\frac{C_i}{S} T_V(t_q) \geq \theta$, where θ is a pre-determined threshold[4]. This strategy assures that all pairs of strings with similarity of at least θ survive the pre-filtering stage and put into the candidate set with a desired probability, as long as the proper sample size S is given.

[3] http://www.doi.org/

[4] In experimentation, we used the more optimized version of the sampling-based join approximation with a single scan from [11].

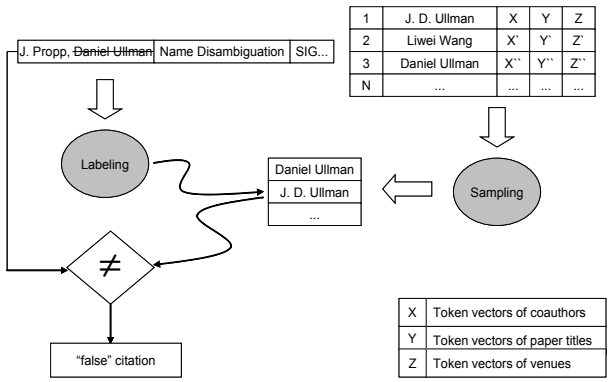


Figure 3: Overview of our solution to MC problem.

3. THE MIXED CITATION PROBLEM

3.1 Problem Definition & Solution Overview

Problem Definition. We formally define the *Mixed Citation* problem as follows:

> *Given a collection of citations, C, by an author, a_i, can we quickly and accurately identify false citations by another author a_j, when a_i and a_j have the identical name spellings?*

The challenge here is that since two different authors, a_i and a_j, have the "same" name spellings, one cannot easily distinguish the two by using distance between their names (e.g., Edit distance). To overcome this difficulty, we propose to exploit author's associated information. That is, given an author a_i, we may use additional information such as her coauthor list, common keywords that she often use in the titles of articles, or common publication outlets, etc.

Solution Overview. Consider a citation c_i with a set of coauthors $A = \{a_1, ..., a_n\}$, a set of keywords from the title $T = \{t_1, ..., t_m\}$, and a venue name V. Then, after removing the i-th coauthor a_i ($\in A$), can we correctly label c_i to a_i? That is, when a_i is removed from the citation c_i, can we guess back the removed author using associated information? Let us call this method as *Citation Labeling* algorithm. If we assume that there is a "good" citation labeling function $f_{cl} : c_i \rightarrow a_j$. Then, using the f_{cl}, the original MC problem can be solved as follows. Given citations C by an author a_1:

for each citation c_i ($\in C$)
  remove a_1 (i.e., original name) from coauthor list of c_i;
  f_{cl} is applied to get a_2 (i.e., guessed name);
  if $a_1 \neq a_2$, then c_i is a false citation;
  remove c_i from C;

At the end, C has only correct citations by a_1. Therefore, if one can find a good citation labeling function f_{cl}, then one can solve the MC problem.

3.2 Citation Labeling Algorithm

Let us examine f_{cl} more closely. Suppose one wants to "label" a collection of citations, C, against a set of possible authors A. A naive algorithm, then, is (let ϕ be a similarity measure between a citation c_i and an author a_j):

for each citation c_i ($\in C$)
  examine all names a_j($\in A$);
  return a_j ($\in A$) with MAX ϕ;

This baseline approach presents two technical challenges: (1) Since c_i and a_j are two different entities to compare in real world, the choice of good similarity measure is critical; and (2) When a DL has a large number of citations and authors in the collection, the baseline approach with a time complexity $O(|C||A|)$ is prohibitively expensive to run (e.g., the DBLP has about 0.56 million authors). In order to address these challenges, we propose two solutions as follows.

Similarity between Citation and Author. In [12], authors reported a promising result by representing a citation as 3-tuple of coauthors, titles, and venues. Although proposed for a different problem, the idea of 3-tuple representation of citations can be adapted to our context as follows: the similarity between a citation c and an author a (hereafter, $sim(c, a)$) can be estimated as the similarity between a 3-tuple representation of c and that of a:

$$sim(c, a) = \alpha\ sim(\vec{c_c}, \vec{a_c}) + \beta\ sim(\vec{c_t}, \vec{a_t}) + \gamma\ sim(\vec{c_v}, \vec{a_v})$$

where $\alpha + \beta + \gamma = 1$ (i.e., weighting factors), $\vec{c_c}$, $\vec{c_t}$, and $\vec{c_v}$ are token vectors of coauthors, paper titles, and venues, respectively, of the citation c, and $\vec{a_c}$, $\vec{a_t}$, and $\vec{a_v}$ are token vectors of coauthors, paper titles, and venues from "all" citations of the author a, respectively. In turn, each similarity measure between two token vectors can be estimated using the standard IR techniques such as the *cosine similarity*, $\cos(\theta) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \cdot \|\vec{w}\|}$, along with TFIDF.

For instance, a citation c "E. F. Codd: A Relational Model of Data for Large Shared Data Banks. Commun. ACM 13(6): 377-387 (1970)" is represented as: $\vec{c_c}$ = ["E.F. Codd"], $\vec{c_t}$ = ["Relational", "Model", "Data", "Large", "Shared", "Data", "Banks"], and $\vec{c_v}$ = ["Commun.", "ACM"])[5]. Similarly, an author "John Doe" with two citations ("John Doe, John Smith: Data Quality Algorithm, IQIS, 2005", and "Dongwon Lee, John Doe, Jaewoo Kang: Data Cleaning for XML, ACM/IEEE Joint C. on Digital Libraries, 2005") is represented as: $\vec{a_c}$=["John Doe", "John Smith", "Dongwon Lee", "Jaewoo Kang"], $\vec{a_t}$=["Data", "Quality", "Algorithm", "Cleaning", "XML"], and $\vec{a_v}$=["IQIS", "ACM/IEEE", "Joint", "C.", "Digital", "Libraries"]. In Section 5, we study the variance of handling duplicate tokens (in set and bag models). Then, the similarity of the citation c and an author "John Doe" is equivalent to: $sim(c, a)$. That is, if $sim(c, a)$ is beyond some threshold, we "guess" that c is a false citation and should have been labeled under "John Doe", not "E. F. Codd" (false positive case). When there are many such authors, we label c as the author with the maximum $sim(c, a)$.

Speed-up through Sampling. In general, the baseline approach has a quadratic time complexity which is too expensive for large-size DLs. However, note that for a citation c, one does not need to check if c can be labeled as an author a for all authors. If one can quickly determine candidate author set from all authors (i.e., pre-filtering), then c better be tested against only the authors in candidate set. We use the Gravano et al.'s approximate join algorithm introduced in Section 2 for the pre-filtering. That is,

for each citation c_i ($\in C$)

[5]We pre-prune all stopwords from the title.

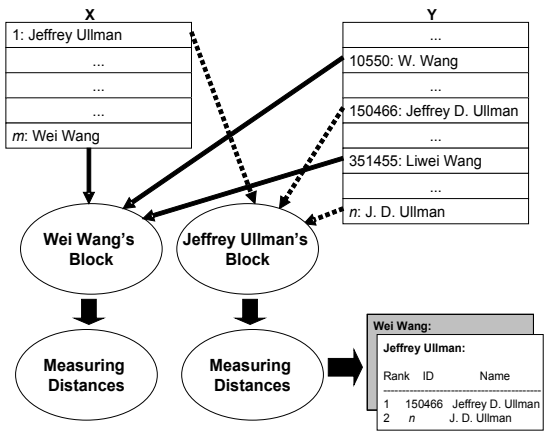


Figure 4: Overview of our solution to SC problem.

draw a sample set $S(\subseteq A)$;
examine all names $s_j(\in S)$;
return s_j $(\in S)$ with MAX ϕ;

Note that the complexity is reduced to $O(|A|+|C||S|)$, that is typically more scalable than $O(|C||A|)$ since $|S| \ll |A|$.

4. THE SPLIT CITATION PROBLEM

4.1 Problem Definition & Solution Overview

We formally define the *Split Citation* problem as follows:

> *Given two lists of author names, X and Y, for each author name x $(\in X)$, find name variants of x: y_1, y_2, ..., y_n $(\in Y)$.*

The baseline approach to solve the problem is to treat each author name as a "string", and perform all pair-wise string distance using some distance function, $dist(x, y)$:

for each name x $(\in X)$
  for each name y $(\in Y)$
    if $dist(x, y) < \phi$, x and y are name variants;

This baseline approach has the limitations similar to the baseline approach of the MC problem in Section 3. Since the baseline approach is prohibitively expensive to run for large DLs (because of its quadratic time complexity, $O(|X||Y|)$)), there is a need for more *scalable* algorithms. Furthermore, many authors from similar cultural or national background shares similar spellings in their name, those algorithms should not be too much dependent on the syntactic similarities of author-name strings.

Solution Overview. Figure 4 illustrates our name disambiguation algorithm: (1) Instead of syntactically comparing two name spellings alone, we use information associated with the author-name strings like coauthor list, authors' paper titles, and venue list. For instance, to identify if "Dongwon Lee" is the name variant of "D. Lee", instead of computing the string edit distance of two names, we may test if there is any correlation between the coauthor lists of "Dongwon Lee" and "D. Lee." In the remainder of the paper, we only focus on exploiting coauthor information as the only associated information of an author. Exploiting other associated information (or even hybrid of them as in [12]) is an interesting direction for future work; (2) To make the algorithm scalable, we again borrow the sampling idea.

Method		Step 1	Step 2
naive	1-N	–	name
two-step name-name	2-NN	name	name
two-step name-coauthor	2-NC	name	coauthor
two-step name-hybrid	2-NH	name	hybrid

Table 1: Solution space of name disambiguation algorithm.

4.2 Name Disambiguation Algorithm

The name disambiguation algorithm is as follows:

/* let C_a be coauthor information of author a; */
for each name x $(\in X)$ draw a sample set $S_x(\in S)$;
for each name y $(\in Y)$ /* Step 1 */
  assign y to all relevant samples $S_i(\in S)$;
for each sample S_x $(\in S)$ /* Step 2 */
  for each name z $(\in S_x)$
    if $dist(C_x, C_z) < \phi$, x and z are name variants;

Note that the time complexity after sampling becomes $O(|X|+|Y|+\mathcal{C}|S|)$, where $\mathcal{C}$ is the average number of names per sample. In general $\mathcal{C}|S| \ll |X||Y|$.

Depending on the choices in both Step 1 and 2, four variations of the name disambiguation algorithm are feasible, as summarized in Table 1: (1) 1-N is a single-step pair-wise name matching scheme without using sampling or coauthor information (i.e., it uses plain pair-wise author name comparison); (2) 2-NN uses the two-step approach, but do not exploit "coauthor" information; (3) 2-NC is the main proposal of ours using the sampling and exploiting "coauthor" information instead of author names; and (4) 2-NH is the modification of 2-NC in that in step 2, it combines both author and coauthor information together with proper weights (e.g., we used 1/4 and 3/4 for author and coauthor, respectively).

Although using the sampling speeds up the whole processing significantly, another important issues is to find out the right distance metric to use in Step 2. Since each author is represented as a potentially very long coauthor list, different distance metrics tend to show different accuracy/performance trade-off. To examine this issue, we have considered two supervised methods (i.e., Naive Bayes Model and Support Vector Machine) and five unsupervised methods (i.e., cosine, TFIDF, Jaccard, Jaro and JaroWinkler). In what follows, we briefly describe each method.

Naive Bayes Model (NBM). In this method, we use Bayes' Theorem to measure the similarity between two author names. For instance, to calculate the similarity between "Dongwon Lee" and "Lee, D.", we estimate the probability per coauthor of "Dongwon Lee" in terms of the Bayes rule in training, and then calculate the posterior probability of "Lee, D." with the coauthors' probability values of "Dongwon Lee" in testing. As shown in Figure 4, given a block corresponding to an author name x in X with the associated author names y_i in Y ($i \in [1, k]$, where k is the total number of author names from Y), we calculate the probability of each pair of x and y_i and find the pair with the maximal posterior probability as follows:

For training, a collection of coauthor names of x are randomly split, and only the half is used for training. We estimate each coauthor's conditional probability $P(A_m|x)$

Name	Description
x, y	coauthor names
T_x	all tokens of the coauthor x
C_x	all characters of x
$CC_{x,y}$	all characters in x common with y
$X_{x,y}$	# of transpositions of char. in x relative to y

Table 2: Terms.

conditioned on the event of x from the training data set, $A_i \in \{A_1, ..., A_j, ..., A_m\}$ and A_j is the j-th coauthor of x:

$$\begin{aligned} P(A_j|x) \quad = \quad & P(A_j|Frequent, Coauthor, x) \times \\ & P(Frequent|Coauthor, x) \times P(Coauthor|x) + \\ & P(A_j|Infrequent, Coauthor, x) \times \\ & P(Infrequent|Coauthor, x) \times P(Alone|x) \end{aligned}$$

where $P(Alone|x)$ is the probability of x writing a paper alone, $P(Coauthor|x)$ is the probability of x working for a paper with coauthors in future, $P(Frequent|Coauthor, x)$ is the probability of x working for a paper with the coauthors, who worked with x at least twice in the training data, conditioned on the event of x's past coauthors, and $P(A_j|Frequent, Coauthor, x)$ is the probability of x working for a paper with a particular coauthor A_j

For testing, we use the following target function: $V_{NBM} = MAX_{y_i \in N}\{P(y_i)\Pi_k P(A_k|y_i)\}$, where N denotes is the total number of author names from Y in the block and A_k is the k-th coauthor in y_i, being the same coauthor as in x.

Support Vector Machines (SVM) is one of the popular supervised classification methods. In our context, it works as follows: First, all coauthor information of an author in a block is transformed into vector-space representation. Author names in a block are randomly split, and 50% is used for training, and the other 50% is used for testing. Given training examples of author names labeled either YES (e.g., "J. Ullman" and "Jeffrey D. Ullman") or NO (e.g., "J. Ullman", "James Ullmann"), the SVM creates a maximum-margin hyperplane that splits the YES and NO training examples. In testing, given the SVM classifies vectors by mapping them via kernel trick to a high dimensional space where the two classes of equivalent pairs and different ones are separated by a hyperplane, and the corresponding similarity is obtained. For the SVM prediction, we use the Radial Basis Function (RBF) kernel [7], $K(x_i, y_i) = e^{-\gamma||x_i - y_i||^2}, (\gamma > 0)$, among alternatives (e.g., linear, polynomial, sigmoid kernels).

String-based Distance Metrics. In this scheme, the distance between two author names are measured by the distance between their coauthor lists (thus no training is needed). That is, to measure the distance between "Dongwon Lee" and "Lee, D.", instead of computing *dist*("Dongwon Lee", "Lee, D."), we compute *dist*(coauthor-list("Dongwon Lee"), coauthor-list("Lee, D.")). Among many possible distance measures, we used two token-based string distances (e.g., *Jaro* and *TFIDF*), and two edit-distance-based ones (e.g., *Jaro* and *JaroWinkler*) that were reported to give a good performance for the general name matching problem in [6]. We briefly describe the metrics below. For details of each metric, refer to [6].

Using the terms of Table 2, the four metrics can be defined as follows: (1) $\mathbf{Jaccard}(\mathbf{x}, \mathbf{y}) = \frac{|T_x \bigcap T_y|}{|T_x \bigcup T_y|}$; (2) $\mathbf{TFIDF}(\mathbf{x}, \mathbf{y}) = \sum_{w \in T_x \cap T_y} V(w, T_x) \times V(w, T_y)$, where $V(w, T_x) = log(TF_{w,T_y} + 1) \times \frac{log(IDF_w)}{\sqrt{\sum_{w'}(log(TF_{w,T_y}+1) \times log(IDF_w))}}$ (symmetrical for $V(w, T_y)$), where TF_{w,T_x} is the frequency of w in T_x, and IDF_w is the inverse of the fraction of names in a corpus containing w; (3) $\mathbf{Jaro}(\mathbf{x}, \mathbf{y}) = \frac{1}{3} \times (\frac{|CC_{x,y}|}{|C_x|} + \frac{|CC_{y,x}|}{|C_y|} + \frac{|CC_{x,y}| - X_{CC_{x,y}, CC_{y,x}}}{2|CC_{x,y}|})$; (4) $\mathbf{JaroWinkler}(\mathbf{x}, \mathbf{y}) = Jaro(x, y) + \frac{max(|L|,4)}{10} \times (1 - Jaro(x, y))$, where L is the longest common prefix of x and y.

Vector-based Cosine Distance. In this approach, instead of using string distances, we use vector distances to measure the similarity of the coauthor lists. We model the coauthor lists as vectors in the vector space, each dimension of which corresponds to a unique author name appearing in the citations. To measure the distance between two vectors, v and w, we use the simple cosine distance, an angle between two vectors, defined as: $\cos\theta = \frac{v \cdot w}{\|v\| \cdot \|w\|}$.

5. EXPERIMENTAL VALIDATION

5.1 Data Sets and Platform

We have gathered real citation data from four different domains, as summarized in Table 3. Compared to previous work, all of the four data sets are substantially "large-scale" (e.g., DBLP has 360K authors and 560K citations in it). Different disciplines appear to have slightly different citation policies and conventions. For instance, Physics and Medical communities seem to have more number of coauthors per article than Economics community. Furthermore, the conventions of citation also vary. For instance, citations in e-Print use the first name of authors as only initial, while ones in DBLP use full names. All four data sets are presegmented (i.e., each field of coauthors, title, and venue are already known to us).

For the sampling technique, we used the implementation of [11] with a sample $S = 64$ and a threshold $\theta = 0.1$. For the supervised learning methods, citations per author are randomly split, with half of them used for training, and the other half for testing. For the implementation of Support Vector Machines, LIBSVM[6] was used. For the string-based distance functions of the unsupervised learning methods, we used the implementations of TFIDF, Jaccard, Jaro, and JaroWinkler from SecondString[7]. Other remaining methods were implemented by us in Java. All experimentation was done using Microsoft SQL Server 2000 on Pentium III 3GHZ/512MB.

5.2 Results for MC Problem

Configuration. For this MC problem, we used two DLs out of four (due to time constraint) as test-beds: DBLP and EconPapers. For DBLP (which authors know well), we collected real examples with the MC problem: e.g., Dongwon Lee, Chen Li, Wei Liu, Prasenjit Mitra, and Wei Wang, etc, and for EconPapers (which authors do not know well), we injected an artificial "false citations" into each author's citation collection. For both data sets, we tested how to find the "false citations" from an author's citations (that is, we had a solution set for both cases). In constructing token vectors, we used two models, *Set* and *Bag*, depending on

[6] http://www.csie.ntu.edu.tw/∼cjlin/libsvm/
[7] http://secondstring.sourceforge.net/

Data set	Domain	# of authors/ # of citations	# of coauthors per author (avg/med/std-dev)	# of tokens in coauthors per author (avg/med/std-dev)
DBLP	CompSci	364,377/562,978	4.9/2/7.8	11.5/6/18
e-Print	Physics	94,172/156,627	12.9/4/33.9	33.4/12/98.3
BioMed	Medical	24,098/6,169	6.1/4/4.8	13.7/12/11.0
EconPapers	Economics	18,399/20,486	1.5/1/1.6	3.7/3/4.1

Table 3: Summary of data sets.

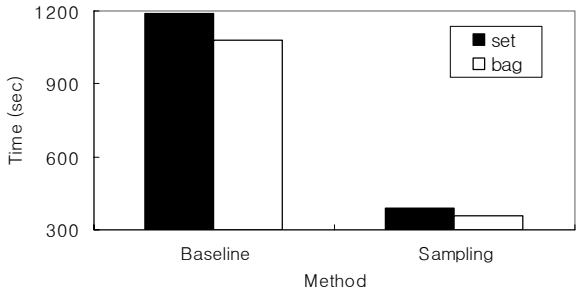
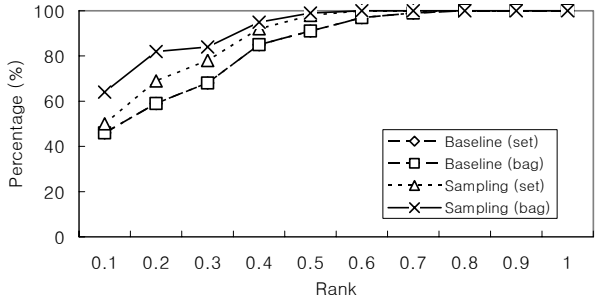
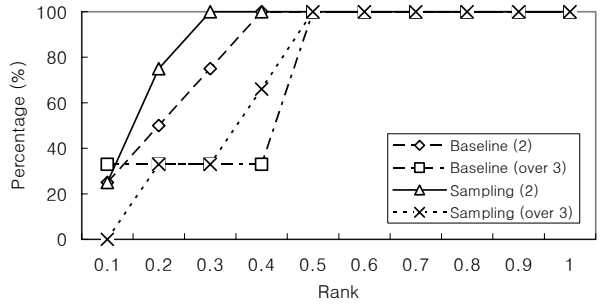


Figure 5: Scalability (EconPapers).



(a) EconPapers



(b) DBLP

Figure 6: Accuracy (EconPapers and DBLP).

the preservation of the multiple occurrences of the same token. For testing, we used the weights, $\alpha = 0.5$, $\beta = 0.3$, and $\gamma = 0.2$. As evaluation metrics, we used *time* for scalability, and *percentage/rank* ratio for accuracy (i.e., A false citation c_f must be ranked low in $sim(c_f, a)$. Thus, we measured how much percentage of false citations were ranked in the bottom 10%, 20%, etc).

Results. First, Figure 5 clearly shows the superior scalability of the sampling-based approach over the baseline one (about 3-4 times faster), regardless of set or bag models. Since the time complexity of the sampling-based approach is bounded by S, which was set to 64, for a large C such as DBLP, the scalability gap between two approaches further widens. Second, Figure 6(a) illustrates the accuracy of both approaches for EconPapers. For instance, when there is a single false citation c_f hidden in the 100 citations, the sampling approach with the bag model can identify c_f with over 60% accuracy (i.e., rank=0.1/%=64). Furthermore, when it can return upto 2 citations as answers, its accuracy improves to over 80% (i.e., rank=0.2/%=82). Since many tokens in citations tend to co-occur (e.g., same authors tend to use the same keywords in titles), the bag model that preserves this property performs better. Finally, Figure 6(b) shows results on DBLP using only the bag model. Note that some collection has a mixture of "2" authors' citations while others have that of "over 3" authors (e.g., there exists more than 3 authors with the same spellings of "Wei Liu"). Intuitively, collections with more number of authors' citations mixed are more difficult to handle. For instance, when 2 authors' citations are mixed, 100% of false citations are always ranked in the lower 30% (i.e., rank=0.3) using the sampling approach. However, when more than 3 authors' citations are mixed, the percentages drop to mere 35% – it is very difficult to decipher a false citation when it is hidden in a collection that contains a variety of citations from many authors. We leave a solution to remedy this problem as a future work.

5.3 Results for SC Problem

Configuration. Ideally, it would be desirable to apply our framework to existing DLs to find all real name variants. However, given a large number of citations that we aim at, it is not possible nor practical to find a "real" solution set. For instance, to determine if "A" is indeed a name variant of "B", human experts have to trace it carefully. Therefore, here, we use synthetic solution sets. Nevertheless, in practice, we envision that our framework be used as a tool to assist human experts to narrow down candidate name variants significantly.

To make solution sets, for each data set, we prepare two initially-empty lists, X and Y. Then, we pick 100 authors with substantial number of citations (so that supervised methods can be trained), and put them into X. For each of 100 original names, x_n, in addition, we create an artificial name variant, y_n. Furthermore, citations of each x_n are randomly split into two sets, and assigned to x_n and y_n, respectively (i.e., each name carries half of the original citations). Finally, to make the disambiguation more challenging, we dump the entire author names into Y. For instance, for DBLP test case, there are 100 and 364,377 names in X and Y, respectively. Then, through the proposed two-step name disambiguation algorithm, for each name in X, we test if the algorithm is able to find the corresponding "artificial" name variant in Y (that we generated and thus know what it is).

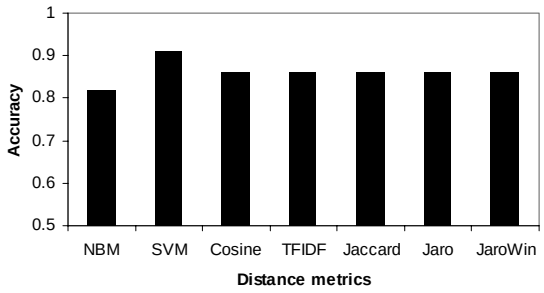
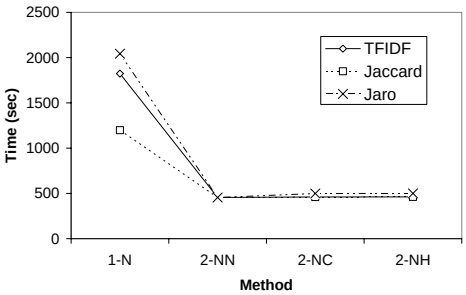


Figure 7: Accuracy (DBLP).

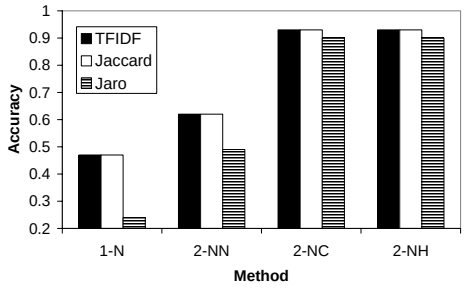
Note that the way we generate artificial name variants may affect the performance of sampling. In general, it is difficult to precisely capture the right percentages of different error types in author name variants. For the original name "Ji-Woo K. Li", for instance, some of possible error types are name abbreviation ("J. K. Li"), name alternation ("Li, Ji-Woo K."), typo ("Ji-Woo K. Lee" or "Jee-Woo K. Lee"), contraction ("Jiwoo K. Li"), omission ("Ji-Woo Li"), or combinations of these. To quantify the effect of error types on the accuracy of name disambiguation algorithms, we first compared two cases: (1) mixed error types of abbreviation (30%), alternation (30%), typo (12% each in first/last name), contraction (2%), omission (4%), and combination (10%); and (2) abbreviation of the first name (85%) and typo (15%). The accuracy of the former case is shown in Figure 7, and that of the latter case is in Figure 10(a). Note that regardless of the error types or their percentages, both cases show reasonably similar accuracies for all seven distance metrics (i.e., 0.8–0.9 accuracy). Therefore, all subsequent experimentations are done using the latter case (85%/15%).

Evaluation metrics. To measure how effectively name variants can be found, we measured the "accuracy" of top-k as follows. For a name in X, our algorithm finds top-k candidate name variants in Y. If the top-k candidates indeed contain the solution, then it's *match*. Otherwise, it is a *mismatch*. This is repeated for all 100 names in X. Then, overall accuracy is defined as: Accuracy $= \frac{\#\text{ of matches}}{100}$. The accuracy was measured for different k values (i.e., $k = 1, 5, 10$). For instance, with $k = 5$ in the DBLP data set, for each author in X, methods return the top-5 candidate name variants out of 364,377 authors, and if one of these 5 candidates is the artificial name variant that we created, then it is a match. We repeated all of the subsequent experiments for three window sizes of 1, 5, and 10, and found that accuracies with larger window size ($k = 10$) are about only 10% higher than those with smaller window size ($k = 1$). Since the different is small, in what follows, we present the results using $k = 5$.

Results. Figure 8 summarizes the experimental results of four alternatives using three representative metrics – TFIDF, Jaccard, and Jaro. In terms of the processing time, 1-N is the slowest for TFIDF and Jaccard, as expected, due to its quadratic time complexity (i.e., $100 \times 364,377$ times of pair-wise name comparisons). The other three show similar performance thanks to the sampling. In terms of accuracy, both 2-NC and 2-NH shows about 20%-30% improvement, compared to 1-N and 2-NN, validating the assumption that



(a) Scalability



(b) Accuracy

Figure 8: DBLP with $k = 1$.

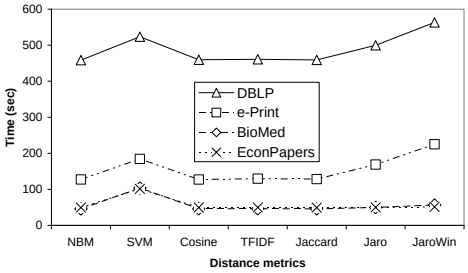


Figure 9: Processing time for Step 2.

exploiting additional information (i.e., coauthor) than the simple name spelling is beneficial. Since 2-NH shows no noticeable improvements over 2-NC, in the remaining experiments, we use 2-NC as a default scheme.

Next, we measured the processing time for step 2 alone (distance measure stage) as shown in Figure 9. In general, token-based distance metrics (e.g., TFIDF, Jaccard) outperforms edit distance based metrics (e.g., Jaro, JaroWinkler). This becomes clearly noticeable for DBLP, but not for EconPapers for its small size. In addition, SVM tends to take more time than the others since the hyperplane needs to be split in succession due to SVM's binary-classifiers.

Figure 10 summarizes the accuracies of our proposal for all four data sets (with $k = 5$). In general, the distance metrics such as the SVM, cosine, TFIDF and Jaccard perform much better than the others. For DBLP data set, most distance metrics achieved upto 0.93 accuracy, finding most of 100 name variants out of 364,377 candidates. For e-Print data set, the accuracy drops down, except the SVM, and for BioMed data set, it gets worse (especially for Jaro and JaroWinkler).

The accuracies of DBLP and e-Print data sets are better than that of BioMed data set. The poor performance of BioMed case is mainly due to the small number of citations

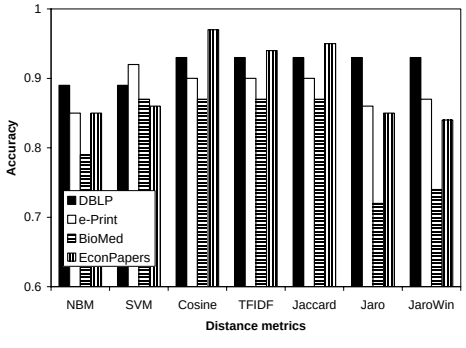


Figure 10: Accuracy ($k = 5$).

per authors in data set. Since 2-NC scheme is exploiting coauthor information of the author in question to find name variants, the existence of "common" coauthor names is a must. However, in the BioMed data set, each author has only a small number of citations, 1.18, on average, and only small number of coauthors, 6.1, on average, making a total number of coauthors as $7.19 = 1.18 \times 6.1$ (assuming all coauthors are distinct). Therefore, for two arbitrary author names x and y, the probability of having "common" coauthors in BioMed data set is not high. On the other hand, for the e-Print data set, the average number of citations (resp. coauthors) per author is higher, 4.27 (resp. 12.94), making a total number of coauthors as $55.25 = 4.27 \times 12.94$ – roughly 8 times of the BioMed data set.

In general, Jaro or JaroWinkler method in step 2 gave poorer accuracy than the others. Since they are edit-distance based methods that are heavily affected by the number of transpositions, as the length of string to compare increases (in 2-NC, it is a long coauthor string to compare), its error rate increases as well. In the e-Print data set, the accuracies are lower, compared to those of DBLP. This is because most of citations in e-Print data set use abbreviation for the first name of authors. Since the sampling technique uses TFIDF for weighting tokens, common tokens like abbreviated first name (e.g., "E." or "P.") would have lower weight via IDF, negatively affecting matching process.

6. CONCLUSION

Two interesting and practical problems – *Mixed Citation* and *Split Citation* – are formally introduced and their solutions are explored. Since both problems commonly occur in many of the existing bibliographic digital libraries, it is important to devise effective and efficient solutions to them. By utilizing one of the state-of-the-art sampling-based approximate join techniques, our solutions are scalable yet highly effective. Furthermore, our proposals exploit associated information of author names (e.g., coauthors, titles, or venues) than names themselves, achieving 90-93% accuracy overall.

As to future direction, in addition to comparing ours against others (e.g., [2, 22]), we plan to apply our framework to other domains (e.g., address, movie) to test its generality.

7. REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. "Eliminating Fuzzy Duplicates in Data Warehouses". In *VLDB*, 2002.

[2] I. Bhattacharya and L. Getoor. "Iterative Record Linkage for Cleaning and Integration". In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2004.

[3] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. "Adaptive Name-Matching in Information Integration". *IEEE Intelligent System*, 18(5):16–23, 2003.

[4] V. R. Borkar, K. Deshmukh, and S. Sarawagi. "Automatic Segmentation of Text into Structured Records". In *ACM SIGMOD*, 2001.

[5] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. "Robust and Efficient Fuzzy Match for Online Data Cleaning". In *ACM SIGMOD*, 2003.

[6] W. Cohen, P. Ravikumar, and S. Fienberg. "A Comparison of String Distance Metrics for Name-matching tasks". In *IIWeb Workshop held in conjunction with IJCAI*, 2003.

[7] N. Cristianini and J. Shawe-Taylor. *"An Introduction to Support Vector Machines"*. Cambridge U. Press, 2000.

[8] J. M. B. Cruz, N. J. R. Klink, and T. Krichel. "Personal Data in a Large Digital Library". In *ECDL*, 2000.

[9] P. T. Davis, D. K. Elson, and J. L. Klavans. "Methods for Precise Named Entity Matching in Digital Collection". In *ACM/IEEE JCDL*, 2003.

[10] I. P. Fellegi and A. B. Sunter. "A Theory for Record Linkage". *J. of the American Statistical Society*, 64:1183–1210, 1969.

[11] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. "Text Joins in an RDBMS for Web Data Integration". In *WWW*, 2003.

[12] H. Han, C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. "Two Supervised Learning Approaches for Name Disambiguation in Author Citations". In *ACM/IEEE JCDL*, Jun. 2004.

[13] M. A. Hernandez and S. J. Stolfo. "The Merge/Purge Problem for Large Databases". In *ACM SIGMOD*, 1995.

[14] Y. Hong, B.-W. On, and D. Lee. "System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach". In *ECDL*, 2004.

[15] J. A. Hylton. *"Identifying and Merging Related Bibliographic Records"*. PhD thesis, Dept. of EECS, MIT, 1996. LCS Technical Report MIT/LCS/TR-678.

[16] M. A. Jaro. "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida". *J. of the American Statistical Association*, 84(406), 1989.

[17] R. P. Kelley. "Blocking Considerations for Record Linkage Under Conditions of Uncertainty". In *Proc. of Social Statistics Section*, pages 602–605, 1984.

[18] S. Lawrence, C. L. Giles, and K. Bollacker. "Digital Libraries and Autonomous Citation Indexing". *IEEE Computer*, 32(6):67–71, 1999.

[19] A. McCallum et al. "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching". In *ACM KDD*, 2000.

[20] B.-W. On, D. Lee, J. Kang, and P. Mitra. "Comparative Study of Name Disambiguation Problem using a Scalable Blocking-based Framework". In *ACM/IEEE JCDL*, 2005.

[21] H. Pasula et al. "Identity Uncertainty and Citation Matching". In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

[22] S. Sarawagi and A. Bhamidipaty. "Interactive Deduplication using Active Learning". In *ACM KDD*, 2002.

[23] M. M. M. Synman and M. Rensburg. "Revolutionizing Name Authority Control". In *ACM DL*, 2000.

[24] J. W. Warnner and E. W. Brown. "Automated Name Authority Control". In *ACM/IEEE JCDL*, 2001.

[25] W. E. Winkler and Y. Thibaudeau. "An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 U.S. Decennial Census". Technical report, US Bureau of the Census, 1991.

Approximate Matching of Textual Domain Attributes for Information Source Integration

Andreas Koeller
Dept. of Computer Science
Montclair State University
1 Normal Ave, Montclair, NJ, USA
koellera@mail.montclair.edu

Vinay Keelara
Dept. of Computer Science
Montclair State University
1 Normal Ave, Montclair, NJ, USA
vinay_keelara@yahoo.com

ABSTRACT

A key problem in the integration of information sources is the identification of related attributes or objects across independent sources. Inferring such meta-information from source data (rather than a-priori available meta-data, such as attribute names) is sometimes possible. For example, existing algorithms attempt to integrate information sources by finding patterns such as Inclusion Dependencies (INDs) across them. However, INDs are based on exact set inclusion and are thus very strict patterns that rarely hold across independent real-world databases.

We propose two error-tolerant measures, termed Similarity Score and Distribution Score, that help identify related attributes across two independent databases, based on similarities in their data. Those measures specifically address the problem of identifying semantic relationships between textual attributes of databases that have few or no equal values.

We also present implementations of those measures and some experimental results.

1. INTRODUCTION

The explosion in the number of data sources on the web and the proliferation of a wide variety of diverse databases has demonstrated the wide variations in the representation of real world entities across data sources. These variations or discrepancies in the representation of the same or related real world entities can arise from the different formats used to store the data, differences in syntax, differences in schema or just the autonomy of the sources of information (and the fact that data is entered by humans).

In any large organization, different parts of the organization may use different systems to produce and store their data. This may be because of lack of co-ordination, different rates of adopting new technology or because of mergers and acquisitions, in which the new organization inherits the data of the original entities. It will now have to access information from multiple original data sets as well as consolidate them so as to eliminate duplication. However the original data sets may not be immediately compatible with each other because the data may be stored in different DBMS, in different formats and schemas.

Many organizations now also collect or correlate their data from autonomous sources within the organization as well as from external sources, to perform data warehousing, mining and statistical analysis. All these applications rely on correct information about database relationships to perform their tasks.

Correlations between the data are typically based on common domains or ontologies or existence of a global domain across the sources. If such meta information is lacking, incomplete, or wrong, it can help to detect relationships between sources from the source *data*, rather than schema. While schema-based matching is performed semi-automatically, and significant human input is often necessary, data comparisons have the potential to be executed entirely by algorithms, leading to increased efficiency and potentially new applications in the area of data integration.

This paper addresses the issue of comparing two independent databases (i.e., collections of data objects, each of which has a number of attributes), and discovering sets of related attributes between them. The approach takes into account that data is rarely perfect, that data objects representing the same real-world entity usually do not have equal values, and also tolerates missing or faulty data. It relies on the assumption that data is primarily stored in textual form and that related data objects show at least some textual similarity. At the basic level, it makes use of the well-known TF-IDF score for text comparisons, integrating this score into the novel measures of *Similarity Score* and *Distribution Score*. While the first score measures relatedness between attributes without duplicates, the second score explicitly takes duplicate values in attributes into account. In combination, the two scores give a measure of relatedness of attributes, in the sense that a high Similarity (or Distribution) Score between two attributes suggests that those two attributes likely refer to the same real-world entity type.

Furthermore, our scores are defined on *sets of attributes*, which enables us to compare entire relations and identify sets of related attributes between them. This procedure is akin to the process of inferring functional dependencies [2] or inclusion dependencies [5] solely from database data, but goes beyond those *exact* patterns. Instead, our scores give degrees of confidence in the relationship between attributes,

IQIS 2005, June 17, 2005, Baltimore, MD, USA

making them appropriate for real-world, noisy data.

This paper is organized as follows: Section 2 reviews background material on the related concept of inclusion dependencies, as well as the TF-IDF text comparison metric. Section 3 defines the Similarity and Distribution Scores for pairs of individual attributes and also introduces an algorithm to apply those scores to *sets* of attributes. Section 4 gives a very brief algorithmic analysis, and Section 5 provides some experimental results on effectiveness of our scores and the efficiency of their computation. Section 6 reviews related work and Section 7 gives conclusions and outlook.

2. BACKGROUND

2.1 Schema Matching

Schema matching, that is the identification of semantically related schema elements across independent databases, is an important task that should be automated as much as possible. Two basic approaches exist [22]. In *schema-only matching*, database fields related to each other are identified based only on the schema of the data sources considered. This matching may be implemented for example through linguistic means on schema element names or through constraint matching (e.g., in the ARTEMIS/MOMIS project [3]). In the second approach, *instance-based matching*, one considers the properties of the data contained in the database fields to conclude that two fields are related. Some important instance-based matching projects are SemInt [19] and LSD [10]. Some recent works in database integration involve a combination of both schema level and instance level matching. The *Corpus-based* schema matcher [12] incorporates this approach. A paper by Kang and Naughton [14] proposes an approach at instance-based matching measuring entropy and mutual information across attributes.

A promising direction for instance-based matching is the discovery of *Inclusion Dependencies* (INDs) between databases [16, 9].

DEFINITION 1 (INCLUSION DEPENDENCY). *Assume two relations R and S. An* Inclusion Dependency *is a rule of the form*

$$R\left[\overline{X}\right] \subseteq S\left[\overline{Y}\right]$$

between a set $\overline{X}$ of attributes from R and a set $\overline{Y}$ of attributes from S, with $\left|\overline{X}\right| = \left|\overline{Y}\right|$ (i.e., with the same number of attributes in each set).

If an inclusion dependency (IND) of substantial size is true between two tables R and S, those relations can be considered related to each other. INDs represent subset-patterns across relations [1], which might span multiple attributes. For example, a foreign key constraint between two relations implies an inclusion dependency between the key and foreign key attributes. However, INDs represent *exact* subsets across two relations and are unlikely to be true in real-world data integration scenarios. INDs cannot model slight variations in the representation of data or data extents that overlap only partially.

IND discovery algorithms [16, 9] usually work bottom-up, first discovering INDs between individual attributes, then between pairs of attributes, and so on. While basic levelwise algorithms akin to the *A-priori* strategy used in association rule mining have been proposed [8], the problem's exponential nature requires more complex algorithms [16, 9] as well as heuristic approaches [17]. In general, the discovery of IND (and other) relationships by comparing the values of the fields of the database is a very complex process, and the problem is NP-hard as a function of the number of attributes in the two relations (since there are 2^k possibly related sets of attributes between two relations with k attributes) [5].

While INDs are defined mainly for relational databases, the concepts apply to all data models that have the basic concept of *attribute*.

[1]Note that INDs are not symmetric, i.e., $R[\overline{X}] \subseteq S[\overline{Y}]$ does not imply $S[\overline{Y}] \subseteq R[\overline{X}]$

2.2 Approximate Matching of Textual Data

Table R

X1	X2
SQL Server	Microsoft Labs
Office	Microsoft Marketing
Magic Broom	American Inc Marketing Research Labs
Telephone	ATT Inc
Pentium	Intel Research Labs
Wireless Mouse	Microsoft Inc
Relational DB	Oracle Labs
Free Phone Plan	SBC Inc and SBC Telecom
Personal Computer	IBM Research and Marketing
Passenger Airplane	Boeing Inc
Wireless Phone	ATT Research and ATT Telecom

Table S

Y1	Y2
Wireless Telephone	ATT Telecom Research
SQL Server DB	Microsoft Research
The Pentium Processor	Intel Research Facilities
DB Technology	Oracle Research Labs
DB Technology	Foxpro International Inc
The Magic Broom	American Marketing Research
Almost Free Phone Plan	SBC Telecom and Labs
Calculator Technology	Texas Instruments Labs
Airplane	Boeing Corp
Personal Computer Technology	IBM Inc Marketing
Broom	American Marketing Research

Figure 1: Two tables with Related Data

To motivate the need for a relaxation of the strict Inclusion Dependency idea, consider two attributes from two different relations, as shown in Fig. 1.

Even though neither of these two sets is a subset of the other, the data in the first table is clearly related to the data in the second table when matching attribute $X1$ to $Y1$ and $X2$ to $Y2$, i.e., $R[X1, X2] \approx S[Y1, Y2]$. Algorithms trying to find inclusion dependencies across such databases [16, 9] will fail to detect relationships in this case.

To approach this problem, we adapt results from the Information Retrieval domain. A useful tool from that domain to measure similarity between texts is the TF-IDF (Term Frequency/Inverse Document Frequency) metric [23].

This metric is defined on a collection of *documents*, each of which consists of *words*. In the relational database domain, documents correspond to individual values in a relational attribute. This metric would then apply primarily to attributes in textual domains, whose values can be broken down into words.

The TF-IDF metric defines a distance between individual strings based on the frequency of their words within the string and within the collection of documents (values) in the database [6]. Let C be a collection of documents (corresponding to an *attribute* in the database domain), and T the dictionary of C (i.e., the union of all words used in all documents of C). Then, for each word $t \in T$, we define C_t to be the subset of C whose documents (values) contain the word t. For each document $c \in C$, we also define a corresponding *document vector* $v_c \in \mathbb{R}^{|T|}$, whose components $v_{c,t}$ each correspond to a term $t \in T$. Then, with $TF_{c,t}$ as the count of t in c and $IDF_t = \frac{|C_t|}{|C|}$, the TF-IDF score of a component $v_{c,t} \in v_c$ can be defined [6] as

$$v_{c,t} = \begin{cases} (\log(TF_{c,t}) + 1) \cdot \log(IDF_t) & \text{if } TF_{c,t} \geq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Each vector v_c formed from such components is then normalized to unit length. The *TF-IDF distance* between two documents c_1 and c_2 is now defined as the dot-product[2] between the corresponding unit-length document vectors

$$\delta(c_1, c_2) = v_{c_1} \cdot v_{c_2}.$$

The intuition behind this definition is that the angle between two vectors (whose cosine is computed by the dot product) is a measure of similarity between documents. Lower angles (i.e., *higher* TF-IDF values) mean higher similarity. This metric emphasizes common words between the documents, and similarities in the relative *frequencies* of such words. Furthermore, due to the IDF component in the IF-IDF score, it weighs *rare* terms more heavily than *frequent* terms. This is justified since so-called function words (prepositions, articles, conjunctions) usually carry less "meaning" than the less frequent nouns and verbs. Similar documents therefore are those that *share many important terms*. Two more observations are that documents with distinct sets of words are considered unequal (since their vectors' dot-product is 0) and that the order of the words in the documents is ignored.

3. DISCOVERING DATABASE SIMILARITIES

In this work, we combine the idea of database integration through instance matching with the idea of textual similarity metrics to define measures and algorithms for the integration of databases with textual attributes.

3.1 Measuring Similarity of Attributes

In our approach to integration of heterogeneous databases, we assume that many domains describing real world entity types in databases are natural language text. Taking this further, we see that any textual data within the relations of the database can be seen as a set of individual name constants (also represented in natural language text) which correspond to entities in the real world.

In this scenario, only in very rare cases can we rely on the equality of tuples or attributes to identify similar relations. Therefore, we relax the requirement of value equality and now define the concept of similarity as follows.

Definition 2 (Similarity). *Assume two relations R and S. The* similarity *of two tuples $x \in R$ and $y \in S$ is a measure of confidence in the co-reference between them. Two tuples are said to be co-referent if they refer to or are mapped from the same real world entity. We denote similarity of two tuples x and y by $\text{sim}(x, y)$, with $0 \leq \text{sim}(x, y) \leq 1$. A similarity of 1 means certain co-reference, whereas a similarity of 0 means unrelated semantics of x and y. For a given threshold c, x and y are called* similar *if $\text{sim}(x, y) > c$, denoted by $x \approx y$.*

Determining co-reference simply from data is not always possible, since textual data typically does not carry enough information to decide co-reference (i.e., while "AT&T Research" is probably co-referent with "AT&T Labs", the values "Microsoft Research" and "Microsoft Office" certainly are not co-referent).

In this paper, we use the TF-IDF score for two documents as the similarity score for two tuples.

Definition 3 ((Value) Similarity Score). *Assume two attributes X and Y across two databases. The similarity score between two values $x \in X$ and $y \in Y$ is defined as*

$$\text{sim}(x, y) \equiv \delta(x, y)$$

The document collection *as used in the TF-IDF score is defined as the concatenation of X and Y, i.e,. as a (multi)set containing all values of X and all values of Y, possibly with duplicates. In particular, the* size *of the document collection C is $|C| = |X| + |Y|$. The dictionary T is defined by all distinct terms in all values in C.*

We now extend the notion of similarity to attributes:

Definition 4 (Attribute Similarity). *Assume two relations R and S. The* similarity *between an attribute X from R and an attribute Y from S is a measure of confidence in the* co-reference *between them. Analogously to value similarity, we denote attribute similarity by $\text{sim}(X, Y)$, with $0 \leq \text{sim}(X, Y) \leq 1$.*

We now discuss meaningful ways of computing attribute similarity, i.e., establishing a similarity measure that predicts co-reference of attributes.

3.2 Computing Attribute Similarity through Value Similarity

In order to compute attribute similarity, we combine individual value similarity scores. We begin by defining a vector carrying information about attribute similarity, using value similarity as defined above.

Definition 5 (Attribute Similarity Vector). *Assume two attributes X and Y from two relations R and S. Without loss of generality, furthermore assume that $|R| \leq |S|$. Temporarily fix some order of the tuples in X, for example $x_0, x_1, \ldots, x_{|R|-1}$.*

[2] $\vec{a} \cdot \vec{b} = \sum_{0 \leq i < |a|} a_i b_i$ for unit length vectors.

A
Microsoft Labs
ATT Inc
Oracle Labs
Boeing Inc

B
Microsoft Labs
ATT Inc
Microsoft Labs
Oracle Labs
Oracle Labs
Boeing Inc

C
Microsoft Labs
Oracle Labs
Boeing Inc
ATT Inc
Microsoft Labs
Oracle Labs
SBC Inc
IBM Research
Texas Instruments
Verizon Telecom

Figure 2: Similar attributes with different degrees of relatedness

The components of the attribute similarity vector $V_{X,Y} \in \mathbb{R}^{|R|}$ *are then defined as* $v_i = \max_{y \in Y}(\text{sim}(x_i, y))$ *with* $0 \leq i < |R|$. *That is, for each value (tuple) in attribute* X*, the vector contains the best (highest) similarity score with any value (tuple) in* Y.

Note that all components of this vector are numbers between 0 and 1. A simple measure for attribute similarity can now easily be defined as the average value of the attribute similarity vector's components:

Definition 6 (Similarity Score). *The similarity between two attributes* X *and* Y *is defined as*

$$\text{sim}(X, Y) = \frac{\sum_{v \in V_{X,Y}} v}{\dim(V_{X,Y})}$$

where $\dim(V)$ *is the number of components in vector* V.

Example *With the data as given in Fig. 1, the similarity scores between each pair of attributes are:*

	$X1$	$X2$	$Y1$	$Y2$
$X1$	*1*	*0*	*0.591*	*0*
$X2$		*1*	*0*	*0.712*
$Y1$			*1*	*0*
$Y2$				*1*

This table gives the values for $\text{sim}(X, Y)$ *where the attribute* X *is taken from the* smaller *relation compared to attribute* Y*, as required in the definition.* □

In the absence of duplicate values in the attributes compared, the similarity score gives a good first indication of the attributes' relatedness.

3.3 Relationships between Non-Key Attributes

This paper focuses on inclusion-dependency-like approximate relationships between databases. Such patterns often include individual attribute pairs with duplicate values. Consider the example in Fig. 2.

While both A and B are subsets of the attribute C (in the "relational" sense of the term subset, i.e, after removal of duplicates), intuitively, attribute B, rather than attribute A, has a greater degree of similarity with attribute C. For example, the fact that in attribute B company names repeat but in attribute A there are no duplicates probably means they are used in different contexts (and therefore have different semantics). However, the similarity score as defined above will not capture this difference as is it 1 for all pairings of attributes A, B, and C.

The higher similarity of A to C can be captured by comparing the *value distributions* of the attributes, rather than simply comparing their values. Therefore, we define the concept of *Approximate Bag Inclusion* for attributes, based on the principles of bag (multi-set) semantics [1]. The concept of set membership is extended in bag semantics to *element multiplicities*, denoted by μ. For example, in the multi-set $A = \{1, 1, 2, 3\}$, $\mu(A, 1) = 2$ and $\mu(A, 5) = 0$. Set membership is naturally defined as $a \in A \iff \mu(A, a) \geq 1$. A bag A is included in a bag B if $\forall a \in A : \mu(A, a) \leq \mu(B, a)$.

We adapt this definition to our domain of approximate attribute matching to obtain a *Distribution Score* for the comparison of attributes containing duplicates, based on their value distribution.

First, we define an *approximate multiplicity* accounting for the fact that we do not expect to find exact inclusion of tuples across our source relations, followed by the definition of a distance measure between such attributes.

Definition 7 (Approximate Multiplicity). *Assume an attribute* X *from relation* R*, an attribute* Y *from relation* S*, and a tuple* $x \in X$*.* R *and* S *could be the same relation and* X *and* Y *could be the same attribute. The* approximate multiplicity *of* x *in* Y*, denoted by* $\tilde{\mu}(Y, x)$ *is then defined as the number of tuples* $y \in Y$ *for which* $\text{sim}(x, y) \geq c$*, with* c *some constant between 0 and 1.*

In our experiments, we empirically found appropriate values of the similarity parameter c to be 0.95 if $X \neq Y$ and 0.995 if $X = Y$, reflecting the expectation that values within the same relation are more likely to be identical if they are co-referent, since they are controlled by the same data entry process.

Rather than counting exact duplicates, this method identifies a number of tuples in attribute Y that are similar to a given tuple x from an attribute X. If $X = Y$, the number of similar tuples to a given x *in its own attribute* is computed. Recall that when computing TF-IDF scores across attributes in distinct relations R and S, the dictionary T of terms is the union of the individual dictionaries, and the document collection C is the concatenation of the two attributes.

This definition of multiplicity is heuristic and does not imply an exact classification. In particular, clusters of "similar tuples" can overlap, i.e., tuples can be part of multiple clusters, such that the sum of all multiplicities of tuples in a relation could be substantially higher than the number of tuples in that relation.

Definition 8 (Attribute Distribution Vector). *Assume two attributes* X *and* Y *from two relations* R *and* S*. Without loss of generality, furthermore assume that* $|R| \leq |S|$*. Temporarily fix some order of the tuples in* X*, for example* $x_0, x_1, \ldots, x_{|R|-1}$.

Now, for each $x_i \in X$ *use the value similarity scores to compute* $\max_{y \in Y}(\text{sim}(x_i, y))$ *and record the tuple* $y \in Y$ *for which this maximum was obtained. In the case of duplicate tuples* y *with the maximum score, use a random such tuple.*

The components of the attribute distribution vector $D_{X,Y} \in \mathbb{R}^{|R|}$ *are then defined as*

$$d_i = \begin{cases} \frac{\tilde{\mu}(X, x_i)}{\tilde{\mu}(Y, x_i)} & \text{if } \tilde{\mu}(X, x_i) \leq \tilde{\mu}(Y, x_i) \\ 0 & \text{otherwise} \end{cases}$$

That is, the distribution score reflects how many duplicates of a value exist in one attribute, relative to the duplicates of its best-matching related value in another attribute. With this definition, the components of the distribution vector are between 0 and 1, and are higher if the multiplicity of a tuple in relation R approaches the multiplicity of the corresponding tuple in S (if there is any). If the multiplicity of a tuple in R *exceeds* the multiplicity of its corresponding tuple or if there is no corresponding tuple, the value of the component is 0. This definition reflects the traditional definition of "bag inclusion" (which we use as an analogy to Inclusion Dependencies) and also captures the notion that similar value distributions suggest a higher degree of similarity (i.e., more evidence for co-reference) between attributes than distinct counts. Note that for any x_i that has duplicates in X, the corresponding value d_i appears multiple times in $D_{X,Y}$.

We now use this distribution vector to define a distribution score:

DEFINITION 9 (DISTRIBUTION SCORE). *The distribution score between two attributes X and Y is defined as*

$$\mathrm{dis}(X,Y) = \frac{\sum_{v \in D_{X,Y}} v}{\dim(D_{X,Y})}$$

where $\dim(V)$ *is the number of components in vector* V.

Note that with this definition, the fact that frequent values x_i from X are represented multiple times in $D_{X,Y}$ gives higher weight to those values in the distribution score. This is meaningful since values that appear often in a non-key attribute are probably more important to that attribute, such that their impact on the total score should be higher.

Example *In the data from Fig. 2, similarity scores across each pair of columns are all* 1*, since the basic data values are identical. The distribution scores are as follows:*

	A	*B*	*C*
A	*1*	*3/4*	*3/4*
B		*1*	*1*
C			*1*

Those scores suggest that attributes B and C are closely related, while attribute A is not as closely related to either B or C. □

Analogously to the definition of the similarity score, a high distribution score between two attributes (i.e., close to 1) suggests high confidence in co-reference between them.

3.4 Discovery across Multiple Attributes

The similarity score and distribution score can be used to estimate the probability that two attributes across two relations are *co-referent*, i.e., refer to the same real-world entity. However, if comparing entire relations, each with multiple attributes, it is also interesting to determine whether there are *sets* of attributes across the relations that are co-referent. This notion is also inspired by the Inclusion Dependency concept.

The process of discovering multi-attribute relationships between two relations R and S is carried out in two phases. In the *first phase* we exhaustively compare single attributes from each relation using similarity scores and distribution scores. Analogously to the procedure determining Attribute Distribution Vectors, we compute distribution scores for each pair of attributes and for each attribute X from R find the attribute Y in S with the highest distribution score relative to X.

We then attempt to merge the individual attributes to form multi-attribute patterns. For example, when merging two attributes X_1 and X_2 from R, for each tuple $t \in R$, we "union" the (textual domain) values $t[X_1]$ and $t[X_2]$, obtaining a new, virtual attribute $X_{1,2}$. We merge the corresponding attributes in S and then compute the distribution score between those two new attributes.

Note that due to the way the TF-IDF scores are defined, we can incrementally compute the TF-IDF scores of the merged attribute values from the TF-IDF scores of the individual values, without the need for new database queries.

We merge attributes in decreasing order of their individual distribution scores, stopping when the distribution score of the merged set of attributes falls below a threshold.

Note that this is a greedy algorithm, which does not guarantee an optimal solution. However, this simple strategy showed very good results in our experiments.

4. ALGORITHMIC ANALYSIS

A simple algorithmic analysis follows. Assume two relations R and S, with att_R and att_S the number of attributes in R and S, respectively. Furthermore, let T be the dictionary of all terms in the union of all (textual domain) attributes in R and S.

The most time-consuming procedures are the computation of TF-IDF scores, which requires accessing each tuple in each table once, and computation of the similarity and distribution scores, which compares each tuple in R with each tuple in S. Each individual comparison of two tuples requires multiplying their TF-IDF vectors. While those vectors are conceptually very long (they have $|T|$ elements), they are also very sparse (since each individual document contains only a fraction of all dictionary words), such that time- and space-efficient storage is possible, for example by only storing non-zero components of the vectors. We may assume an average k for the number of distinct words in a value in a relation.

The time complexity for computing individual similarity and distribution scores is then in $O(k \cdot |R| \cdot |S|)$, while the algorithm to compare two entire relations runs in $O(k \cdot att_R \cdot att_S \cdot |R| \cdot |S|)$.

The largest data structures in this environment are the arrays storing individual TF-IDF scores. Their unoptimized size is $|R| \cdot |T|$ and $|S| \cdot |T|$ floating point values, respectively, although as mentioned above, those are sparse arrays and sparse matrix storage can be used. Our algorithms in their current form rely on the ability to store those structures in memory.

5. EXPERIMENTS

5.1 Experimental Setup

Algorithms to compute the Similarity Score and Distribution Score were implemented in Java over relational databases. The testing environment for all our experiments consisted of two machines; one 2.5-GHz Pentium-4 Windows based machine with 2 GB memory running Java 1.4.2 and one 4 Processor 1.2-MHz Sun Fire 880 system with 8 GB

memory running various relational database servers.

Test data was obtained from the UC Irvine KDD Archive (`kdd.ics.uci.edu`), and the US Census 2000 archives (`ftp://ftp2.census.gov/census_2000/datasets`). The datasets (converted into relational tables) are:

Movies: This dataset contains a list of movie names. There is information on actors, casts, directors, rating and remakes etc. Outside of the key fields, missing values and duplicates are common in this dataset. We have used different projections of this relation, to carry out our experiments.

Census: This dataset contains a variety of census data. It consists of about a 1000 tuples and 41 attributes. The dataset is mainly used to test the performance of the algorithms.

From these datasets we generated overlapping subsets, partly through random sampling and partly through selections.

5.2 Experiment 1: Behavior of Distribution Score—Effect of Non-distinct Values

The purpose of this experiment is to determine the effect that non-distinct values (duplicates) in a relation have on the behavior of the distribution score. The average distribution scores obtained in each case will determine whether two relations can be integrated. The algorithm is run on two relations R and S, derived from the Movies dataset. R consists of movie names. For each tuple in R, there is exactly one tuple in S that shares all "meaningful" terms (i.e. terms with high TF-IDF scores), but has a few additional "functional words" (i.e. terms with low TF-IDF scores). The three cases considered are

- All the values in the two relations are distinct.
- There are a few tuples with a random number of duplicates.
- There are very few distinct tuples, with each distinct tuple having a large number of duplicates.

In each of the cases we generate a subset of the relation R, denoted by R', through sampling. The size of the subset R' varied from 100% to 5% of the original relation. This procedure ensures that the *similarity scores* for each of the following experiments are exactly 1.

Case 1: The results of the initial case, where all the values are distinct, are shown in Fig. 3. Since every value of the original Movies relation does exist in relation S, it is detected as a good approximate match by the similarity algorithm despite the presence of other function words. Furthermore, every tuple present in relation R and relation S is distinct, i.e., they have no duplicates. Recall that the distribution score measures the number of duplicates for each tuple in R vs the number of duplicates for the "best matching" tuple in S (see Sec. 3.3). Hence, no matter what the sample size or the tuples present in R, the distribution scores for all the tuples will always be equal to 1. Consequently, the distribution score for the entire relation (the average of the individual distribution scores) will also always be equal to 1. We can then conclude that when all the tuples in a relation are distinct, a high similarity score alone is sufficient to detect the inclusion between the two relations.

Case 2: In this experiment, we wanted to verify the hypothesis that distribution score more accurately measures

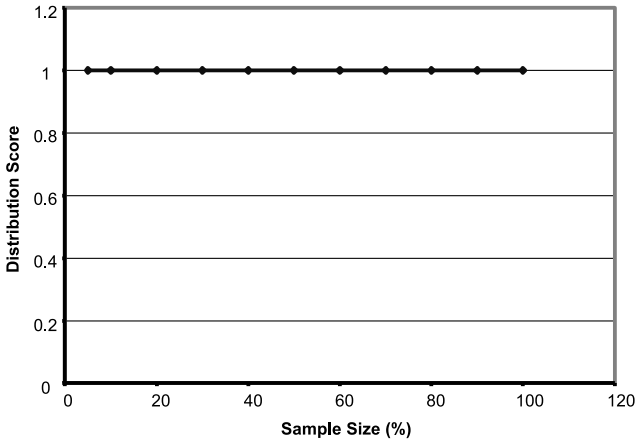


Figure 3: Distribution Scores for relations with no duplicate tuples

"integrability" than similarity score. For this purpose we create a table S with many duplicates and randomly sampled a table R from it. We expect that even if R is small the similarity score will still be high (since all tuples in R still have matches in S), but that the distribution score decreases since the number of duplicates in R now differs from the number of duplicates in S.

The distribution scores of the all tuples without duplicates in R will always be equal to 1 in all the samples, however the distribution scores of the non-distinct tuples will be less than 1 since they now have fewer duplicates of each tuple. Hence the distribution scores of these tuples will tend to decrease as the sample size decreases.

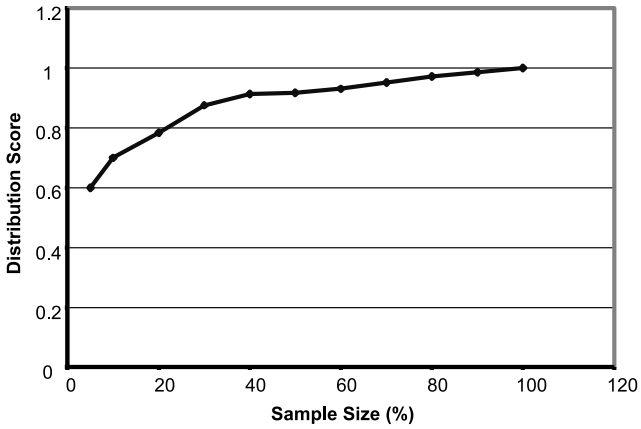


Figure 4: Distribution Scores for relations with the presence of duplicate tuples

As seen in Fig. 4, the distribution score for the entire relation tends to decrease gradually as the sample size decreases, until a certain sample size is reached (sample size of about 35 % for the dataset considered). At this point the scores start to decrease sharply. This is because of two reasons. The sample sizes are now so small that it is less likely that distinct tuples from the original relation (which would produce distribution scores of 1) are included in R. The small sample sizes also mean that non-distinct tuples of S that

are included in R have very few or no duplicates within the sample, leading to lower scores.

This illustrates the fact that even though as in the previous case, every tuple of any sample of S, i.e. any $r \in R$, will have a good approximate match in S, this alone will not give us a good indication of whether the relations can be integrated, since the attributes' value distributions may be different. The use of the distribution score helps in determining if the attributes may be co-referent.

Case 3: In this instance the relations S are formed by a projection on the Rating attribute of the Movies dataset (a textual domain attribute), and relation R is obtained by randomly sampling it. There are only four distinct tuples in the relations, with a large number of duplicates for each tuple. This experiment has been carried out to further emphasize the differences between the results of the similarity score and the distribution score.

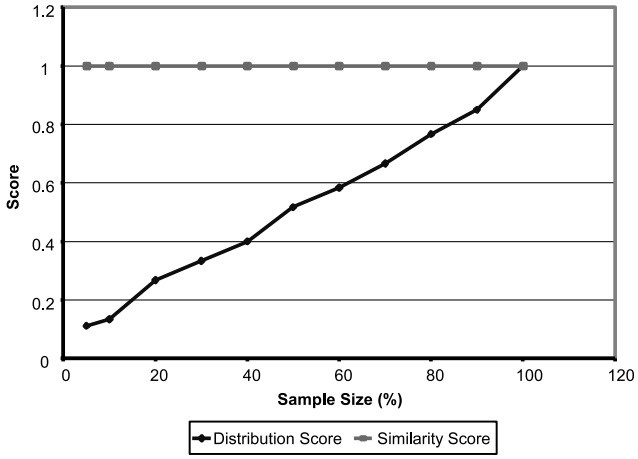


Figure 5: Comparison of Distribution and Similarity scores for relations with low number of distinct tuples.

The results for different samples on relation R are depicted in Fig. 5. Since every value of relation R exists in relation S, all the tuples in any subset of R of any size will always have a good similarity score. However, the distribution scores decrease steadily as the sample size decreases, because the subsets of R will show value distributions less and less similar, with respect to relation S. We see that samples R' of a sample size of 50% of S or less are no longer considered co-referent by our measure, since the distribution of values in relation R' no longer matches the distribution in S. This is not reflected by the by the similarity score at all.

5.3 Experiment 2: Behavior of the Distribution Score in the Presence of Noise due to Unrelated Tuples.

In this experiment we verify the hypothesis that the Distribution Score tolerates noise produced by tuples that do not find good approximate matches, i.e. tuples with low score in the Similarity Score Vector, much better than existing algorithms.

For this experiment we created a relation R from the Movies dataset. The relation S is obtained by removing the desired percentage of tuples from R for each of the test cases considered, ensuring that not all tuples in the "smaller" relation find matches in the "larger" relation S. The comparison is then carried out between a subset sampled from R, denoted by R', and relation S.

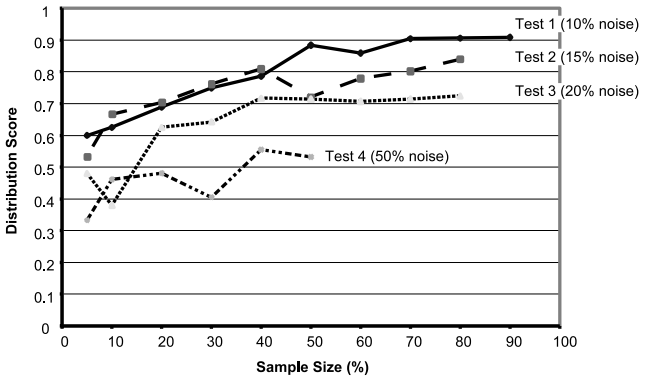


Figure 6: Effect of noise produced by unrelated tuples

We have considered four test cases. In the initial test (*Test 1*), about 10% of the values in R do not find good approximate matches in relation S. This number is progressively increased with each test. In *Test 2* 15%, in *Test 3* 20% and finally in *Test 4* almost 50% of the tuples in R do not find matches in relation S. The experiments are performed on subsets of R of varying sample sizes in each of the test cases.

We see from results shown in Fig. 6 that the Distribution Score is still high (close to 1), indicating co-reference (and thus "integrability") of the two relations at 10% noise level. As we increase the noise level to 15 % and then 20% although the distribution scores decrease correspondingly, the scores are still high enough for the two relations to be considered related. However at 50% noise level there is a significant drop in the distribution scores as indicated by plot for *Test 4* compared to the score obtained for 10% noise level in *Test 1*. Hence at a 50% noise level the relations can no longer be considered related.

This supports our hypothesis that the Distribution Score helps to identify related attributes in noisy relations. As seen from the results of the experiment it detects relationships among database relations at noise levels of up to 20%. Recall here that there is no need for a single tuple to exist in both relations, since TF-IDF scores are used for all comparisons. Rather, the algorithm is tolerant to non-matching data values that have *high* relative TF-IDF scores, as well as to noisy data, i.e., tuples in R that do not even have high TF-IDF scores with any data in S.

The distribution score drops (i.e., suggests lack of co-reference) only at noise levels of almost 50%. This also represents a successful application of the measure, since a very high number of mismatches between the tuples of two relations may indicate that the relations are mapping different real world entities and hence cannot be integrated.

5.4 Experiment 3: Effect of Data Size on Performance

This experiment has been performed to study the relationship between the sizes of the relations considered and the performance of the similarity detection algorithm developed in the dissertation. The experiments have been carried

out in two phases, corresponding to the two phases of multi-attribute pattern discovery outlined in Sec. 3.4. In the first phase we test the performance of the algorithm by varying the tuple sizes of the relations considered. In the second phase the performance of the algorithm is tested by varying number of attributes in the underlying relations.

We have used a projection on a single attribute of the Census dataset to experiment on the first phase (i.e., single attribute comparison). This relation has twenty distinct values. The subsets and superset of this relation were formed by sampling and duplicating the values of the original projection. The results are as shown in Fig. 7.

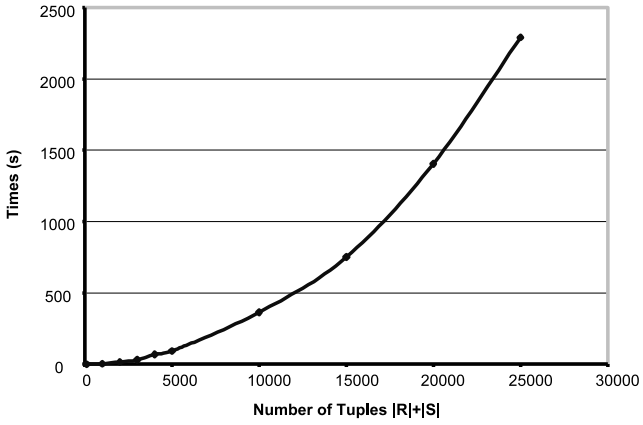


Figure 7: Complexity of Distribution Algorithm with respect to the Number of tuples in the relations

In the dataset considered the number of unique terms in both the relations are constant. Fig. 7 shows that the runtime of the algorithm increases as a quadratic function of sizes of the relations considered ($r^2 = 0.9988$). We increase the sizes of both $|R|$ and $|S|$ by the same ratio to obtain each data point. Since an increase in the size of the relations means an increase in the number of tuples in both the underlying relations, we can conclude that the runtime of the algorithm increases as a linear function of the number of tuples in each of the two individual underlying relations, i.e., as a quadratic function of the sum of the relation sizes.

In implementing the second phase of the discovery (merging attribute for discovery of multi-attribute relationships) we have used subsets formed by projections on the Census dataset. This dataset originally has 41 attributes. The number of attributes considered therefore varies from 1 to 41. The number of tuples in the dataset is 1000 and remains a constant in all the subsets. The results of the experiment are shown in Fig. 8.

Again, the increase in the runtime of the algorithm vs the increase in the size of the relations follows an almost quadratic function ($r^2 = 0.9581$). Since an increase in the size of the relations now means an increase in the number of attributes in both the underlying relations, we can conclude that the runtime of the algorithm increases as a linear function of the number of attributes in each of the two individual underlying relations. The curve is not perfectly quadratic, and the correlation is weaker than in the previous case because of the effect the number of distinct terms has on the runtime. In the dataset considered, the number of distinct terms is not a constant.

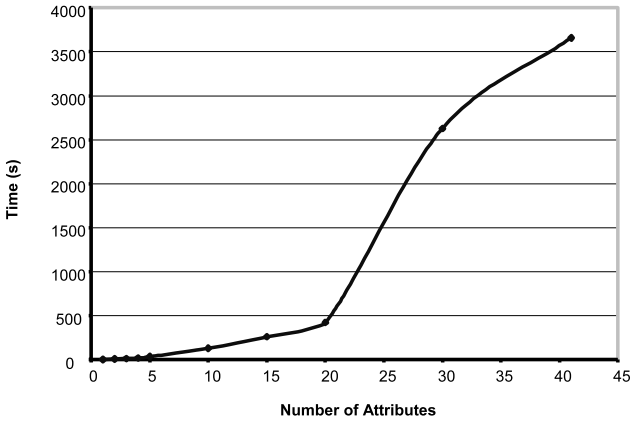


Figure 8: Complexity of Distribution Algorithm with respect to the Number of Attributes in the relations

6. RELATED WORK

The integration of heterogeneous databases is essential in the face of the wide variety of diverse databases currently being used. The problem of finding patterns in databases, and the problem of integrating databases through discovery of schema mappings have both been widely studied [3, 19, 10, 12].

A prominent example for an schema-based matching project is Artemis/MOMIS [3], which attempts to integrate schemas based on attribute name equality, related attribute domains, attribute name synonyms and hypernyms in a schema element hierarchy, and matching of schema sub-structures.

The LSD project [10] integrates a number of different techniques for the discovery of related schema elements. Among those techniques are attribute name similarity measures and instance-driven technique based on attribute or domain values. The LSD project does use a WHIRL-based [6] nearest-neighbor metric for attribute values which, like our Similarity scores, also builds upon the IF-IDF scores. However in LSD, this metric is implemented through a machine-learning algorithm [7], requiring the use of a training set, which is not needed in our work.

Casanova, Fagin & Papadimitriou [5] give a definition of the inclusion dependency problem, prove that it is PSPACE complete, and propose a fundamental axiomatization of IND inference. Mitchell [21]) developed inference rules to derive inclusion dependencies from existing inclusion dependencies and functional dependencies.

de Marchi and Petit [9] as well as Koeller and Rundensteiner [16] have done the most recent work in using the concept of Inclusion Dependency discovery for the integration of databases. Both papers propose algorithms for the detection of inclusion dependency patterns across databases. However, neither paper deals with noise or non-matching values. More recent work [17] addresses the exponential nature of the IND discovery problem by proposing complexity-reducing heuristics, but also does not address the problem of non-equality at the data value level.

Bell and Brockhausen [2], Huhtala et al. [13], Knobbe and Adriaans [15] and Savnik and Flach [11] have contributed

solutions on problems of discovery of functional dependencies in databases. Furthermore, some authors, such as Lim and Harrison [20] or Wei and Chen [25] address the topic of discovering functional-dependency-like patterns that are *almost* true. This topic is related to our approximate inclusion dependency discovery. However, functional dependencies are a special case of inclusion dependencies, in which the "right" side of the dependency is a key in its table, i.e., contains no duplicates. The IND discovery problem is more general, such that functional dependency discovery algorithms are not usable for inclusion dependency discovery.

Work on discovery of similarities in databases is not limited to Inclusion Dependency Discovery. Larson et al. [18] propose a theory of attribute equivalence in databases, based on the domains, the minimal and maximal values of the attribute, and constraints imposed by the database. However they do not examine the data inside the attributes further, making their proposal very sensitive to the availability and correctness of database constraints, as well as the availability of either global or common domains.

Kang and Naughton [14] propose an instance-based technique of examining statistical characteristics of data (specifically, mutual information content of attributes) to identify dependency relationships between data in the attributes of independent tables. This approach is related to ours, but requires equality of entire data values in each attribute, rather than looking at individual words as in our work.

Bilenko and Mooney [4], propose learnable text similarity metrics, to aid in duplicate detection. This approach is unrelated to the duplicate detection mechanism in this paper. It deals mainly with differences in the spelling of text being compared. This feature could in the future be integrated with our similarity measures to increase their applicability.

Schallehn, Sattler, and Saake [24], present an algorithm for data integration, by proposing variants of the SQL grouping and join operators. In fact the authors present an approach that is also based on the concepts of similarity and duplicate detection. However, to determine similarity the authors depend upon *similarity predicates*, which are application specific and user defined. This approach assumes availability of extensive and accurate knowledge about the contents of the tables being considered. This is an assumption we do not make in our work, taking into consideration the real-world applicability of our algorithms.

7. CONCLUSION AND FUTURE WORK

In this paper we have proposed two measures, called *Similarity Score* and *Distribution Score* to solve some problems of data integration of heterogeneous databases. With this approach it is now possible to detect similarities from data in databases even in the presence of a large amount of "noise" in the contents of the database.

The implementation of approximate matching of textual domain attributes addresses the problem of differences in the representation of real world entities in database systems. The application of the distribution score for the integration of databases shifts the focus from discovering Inclusion Dependencies which are defined to be set inclusions, to satisfying more "fuzzy" patterns. Hence, we move away from a binary result (i.e., whether two sets of attributes are related or not), to providing a score that reflects our degree of confidence that attributes are related, i.e., that their relations can be integrated.

The algorithms implemented have a linear runtime with respect to the sizes of the databases as well as the number of attributes in the databases, making them applicable for integration of large database systems and preserving their relevancy for real world applications. The good runtime is largely due to the fact that introducing the similarity and distribution scores for attributes enables us to define a greedy attribute matching algorithm as opposed to an enumeration strategy that is necessary for detection of exact inclusion dependencies.

While the algorithms provided in this paper advance the mechanisms for integration of databases, additional work is necessary before a deployment in the real world can occur. Some of the improvements that are required are

- Handling data types other than strings such as dates, numbers
- Reduction of sensitivity to variations in spelling
- Increasing the efficiency of the algorithms by reducing runtime
- A more comprehensive treatment of value distributions, beyond simple duplicate counts. Mechanisms such as χ^2 independence tests or Kang/Naughton's [14] mutual information content approach might strengthen our results.

8. REFERENCES

[1] J. Albert. Algebraic properties of bag data types. In *International Conference on Very Large Data Bases*, pages 211–219, 1991.

[2] S. Bell and P. Brockhausen. Discovery of data dependencies in relational databases. In Y. Kodratoff, G. Nakhaeizadeh, and C. Taylor, editors, *ML-Net Familiarization Workshop*, 1995.

[3] S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering*, 36(3):215–249, 2001.

[4] M. Bilenko and R. J. Mooney. Employing trainable string similarity metrics for information integration. In *Proc. of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, pages 67–72, Acapulco, Mexico, 2003.

[5] M. A. Casanova, R. Fagin, and C. H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. In *Proceedings of ACM Conference on Principles of Database Systems (PODS)*, pages 171–176, 1982.

[6] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. *SIGMOD Record*, 27(2):201–213, 1998.

[7] W. W. Cohen and H. Hirsh. Joins that generalize: Text classification using WHIRL. In *Proc. of 4th Intl. Conf. on Knowl. Discovery and Data Mining (KDD)*, page 1998, New York, August 169–173.

[8] F. de Marchi, S. Lopes, and J.-M. Petit. Efficient algorithms for mining inclusion dependencies. In *Proceedings of International Conference on Extending Database Technology (EDBT)*, pages 464–476, 2002.

[9] F. de Marchi and J.-M. Petit. Zigzag: A new algorithm for mining large inclusion dependencies in databases. In *3rd Intl. Conf. on Data Mining*, pages 27–34, Melbourne, Florida, November 2003. IEEE.

[10] A. Doan, P. Domingos, and A. Halevy. Learning source description for data integration. In *Proceedings of the Third International Workshop on the Web and Databases (WebDB)*, pages 81–86, Dallas, 2000.

[11] P. A. Flach and I. Savnik. Database dependency discovery: a machine learning approach. *AI Communications*, 12(3):139–160, November 1999.

[12] A. Halevy and J. Madhavan. Corpus-based knowledge representation. In *Proc. of 18th Intl. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 1567–1572, Acapulco, Mexico, 2003. Morgan Kaufman.

[13] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Efficient discovery of functional and approximate dependencies using partitions. In *Proceedings of IEEE International Conference on Data Engineering*, pages 392–401, 1998.

[14] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. *Proceedings of SIGMOD*, pages 205–216, 2003.

[15] A. J. Knobbe and P. W. Adriaans. Discovering foreign key relations in relational databases. In R. Trappl, editor, *Proceedings of the Thirteenth European Meeting on Cybernetics and Systems Research*, volume 2, pages 961–966, Vienna, Austria, 1996. Austrian Soc. Cybernetic Studies, Vienna, Austria.

[16] A. Koeller and E. A. Rundensteiner. Discovery of high-dimensional inclusion dependencies. In *Proceedings of IEEE International Conference on Data Engineering*, pages 683–685, Bangalore, India, 2003. IEEE.

[17] A. Koeller and E. A. Rundensteiner. Heuristic strategies for Inclusion Dependency discovery. In *Proc. of 3rd Intl. Conf. on Ontologies, Databases, and Applications of Semantics (ODBASE'04)*, volume 3290–3291 of *LNCS*, pages 891–908, Larnaca, Cyprus, 2004. Springer.

[18] J. A. Larson, S. B. Navathe, and R. Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, April 1989.

[19] W. Li and C. Clifton. SemInt: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data and Knowledge Engineering*, 33(1):49–84, 2000.

[20] W. Lim and J. Harrison. Discovery of constraints from data for information system reverse engineering. In *Proc. of Australian Software Engineering Conference (ASWEC '97)*, Sydney, Australia, Sep 28–Oct 2 1997.

[21] J. C. Mitchell. Inference rules for functional and inclusion dependencies. In *Proceedings of ACM Symposium on Principles of Database Systems*, pages 58–69, Atlanta, Georgia, 21–23 March 1983.

[22] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[23] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Adison Wesley, New York, 1989.

[24] E. Schallehn, K.-U. Sattler, and G. Saake. Efficient similarity-based operations for data integration. *Data and Knowledge Engineering*, 48(3):361–387, 2004.

[25] Q. Wei and G. Chen. Efficient discovery of functional dependencies with degrees of satisfaction. *Intelligent Systems*, 19(11):1089–1110, September 2004.

Clustering Mixed Numerical and Low Quality Categorical Data: Significance Metrics on a Yeast Example

Bill Andreopoulos
Department of Computer Science, York University, Toronto, Canada, M3J1P3
billa@cs.yorku.ca

Aijun An
Department of Computer Science, York University, Toronto, Canada, M3J1P3
aan@cs.yorku.ca

Xiaogang Wang
Department of Mathematics and Statistics, York University, Toronto, Canada, M3J1P3
stevenw@mathstat.yorku.ca

ABSTRACT

We present the M-BILCOM algorithm for clustering mixed numerical and categorical data sets, in which the categorical attribute values (CAs) are not certain to be correct and have associated confidence values (CVs) from 0.0 to 1.0 to represent their certainty of correctness. M-BILCOM performs bi-level clustering of mixed data sets resembling a Bayesian process. We have applied M-BILCOM to yeast data sets in which the CAs were perturbed randomly and CVs were assigned indicating the confidence of correctness of the CAs. On such mixed data sets M-BILCOM outperforms other clustering algorithms, such as AutoClass. We have applied M-BILCOM to real numerical data sets from gene expression studies on yeast, incorporating CAs representing Gene Ontology annotations on the genes and CVs representing Gene Ontology Evidence Codes on the CAs. We apply novel significance metrics to the CAs in resulting clusters, to extract the most significant CAs based on their frequencies and their CVs in the cluster. For genomic data sets, we use the most significant CAs in a cluster to predict gene function.

1. INTRODUCTION

Clustering aims to partition a set of objects into clusters, so that objects with similar characteristics are clustered together and different clusters contain objects with dissimilar characteristics. A high quality clustering tool produces clusters with *high intra-class* similarity between objects and *low inter-class* similarity between objects [11, 13, 15, 17]. Many numerical data sets have CAs associated with them, but not all CAs are certain to be correct. For many of these data sets CVs can be extracted on the CAs, representing the certainty about the CAs' correctness [8, 20].

We designed the M-BILCOM clustering tool for numerical data sets that incorporates in the clustering process CAs and CVs indicating the confidence that the CAs are correct. M-BILCOM was mainly inspired by numerical gene expression data sets from DNA microarray studies, where CAs and CVs can be derived from Gene Ontology annotations and Evidence Codes [4-10, 12, 14, 21-22]. One of the main advantages of this algorithm is that it offers the opportunity to apply novel significance metrics for spotting the most significant CAs in a cluster when analyzing the results [3]. In genomic data sets, our significance metrics allow significant CAs to be extracted from a cluster based on their CVs and their frequencies and to be used for predicting the functions of other genes in the cluster. This provides a different insight for predicting gene function by giving the 'full picture' of the data set, because the significant CAs are extracted from genes that may have been appended to the cluster on the basis of numerical or categorical similarity or both.

This approach offers several advantages over other approaches:

- Our clustering algorithm may cluster data sets where all genes have numerical attribute values but not all genes have CAs. Each CA has a CV associated - a real number between 0.0 and 1.0 - indicating our confidence about its correctness.
- During the clustering process, this method starts from CAs and CVs at the lower level and then moves to numerical clustering at a higher level. The CAs and CVs are actually used in the clustering process, instead of just annotating the clusters afterwards [1, 3]. The method of Wu et al. as applied previously to high-throughput biological data, starts from the numerical clustering, then adds CAs at a higher level and finally CVs are calculated (P-values) [25].
- During the clustering process, objects having CAs with high confidence to be correct, get clustered by emphasizing more the categorical similarity and less the numerical similarity. On the other hand, objects having CAs with low confidence to be correct get clustered by emphasizing more the numerical similarity [1].
- Our clustering algorithm allows us to define significance metrics indicating the significance of a CA in a cluster. Such metrics are calculated on the basis of how frequently a CA appears in a cluster as well as how strongly the CVs support the CA's correctness in that cluster [3].
- For genomic data sets CVs can be derived from GO evidence codes to point out the most reliable CAs to be used for gene functional prediction purposes [8, 12]. This is in contrast to previous methods, where CVs were calculated at the end to indicate the reliability of a CA's belonging to a cluster [25].

Section 2 describes the k-Modes clustering algorithm. Section 3 discusses the M-BILCOM clustering algorithm, which is a combination of MULICsoft and BILCOM. Section 4 proposes two significance metrics for the CAs in the resulting clusters and discusses their utility for gene functional prediction on a real yeast data set. Sections 5 and 6 describe the results for applying M-BILCOM to highly noisy yeast data sets and its ability to reproduce the correct cluster structure. Sections 7 and 8 discuss implications of the significance metrics for biologists and gene functional prediction. Finally, Section 9 concludes the paper.

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

2. BACKGROUND ON K-MODES

k-Modes is a clustering algorithm that deals with categorical data only [18,19]. The k-Modes clustering algorithm requires the user to specify from the beginning the number of clusters to be produced and the algorithm builds and refines the specified number of clusters. Each cluster has a mode associated with it. Assuming that the objects in the data set are described by m categorical attributes, the mode of a cluster is a vector $Q=\{q_1, q_2, \ldots, q_m\}$ where q_i is the most frequent value for the ith attribute in the cluster of objects.

Given a data set and the number of clusters k, the k-Modes algorithm clusters the set as follows:

1. Select initial k modes for k clusters.
2. For each object X
   a. Calculate the similarity between object X and the modes of all clusters.
   b. Insert object X into the cluster c whose mode is the most similar to object X.
   c. Update the mode of cluster c
3. Retest the similarity of objects against the current modes. If an object is found to be closer to the mode of another cluster rather than its own cluster, reallocate the object to that cluster and update the modes of both clusters.
4. Repeat 3 until no or few objects change clusters after a full cycle test of all the objects.

A similarity metric is needed to choose the closest cluster to an object by computing the similarity between the cluster's mode and the object. Let $X=\{x_1, x_2, \ldots, x_m\}$ be an object, where x_i is the value for the ith attribute, and $Q=\{q_1, q_2, \ldots, q_m\}$ be the mode of a cluster. The similarity between X and Q can be defined as:

$$\text{similarity}(X, Q) = \sum_{i=1}^{m} \delta(x_i, q_i)$$

$$\text{where } \delta(x_i, q_i) = \begin{cases} 1 & (x_i = q_i); \\ 0 & (x_i \neq q_i). \end{cases}$$

Given the similarity measure and k, the clustering result produced by the k-Modes algorithm on a set of data depends on the initial modes and the ordering of the objects presented to k-Modes. In [18] two methods for selecting the initial modes are discussed and compared.

In the descriptions that follow we assume that C represents the total number of clusters and we use c to index the clusters. We assume that m represents the number of attributes in an object of the data set and N represents the number of objects in the data set.

3. THE M-BILCOM CLUSTERING ALGORITHM

M-BILCOM is a combination of MULICsoft [2] and BILCOM [1]. The basic idea of our algorithm is to do clustering at two levels, where the first level clustering imposes an underlying framework for the second level clustering, thus simulating a Bayesian prior as described in [1]. The categorical similarity is emphasized at the first level and the numerical similarity at the second level. *The level one clusters are given as input to level two and the level two clusters are the output of the clustering process.* The process looks as in Figure 1. As shown, both level one and level two involve the same number of clusters, four in this example. The level two clusters consist of subclusters. Data object A was assigned to different level one and level two clusters, because the numerical similarity at the second level was stronger than the categorical similarity at the first level. Thus, in the case of Figure 1 the following relationship holds for object A:

$$categorical\_similarity(A, cluster2) + numerical\_similarity(A, cluster2) >$$

$$categorical\_similarity(A, cluster3) + numerical\_similarity(A, cluster3)$$

On the other hand, data object B was assigned to the same clusters in both levels one and two, because both numerical and categorical similarity supported this classification. Thus, this algorithm considers both categorical and numerical similarity of a data object to the clusters to which it may be allocated.

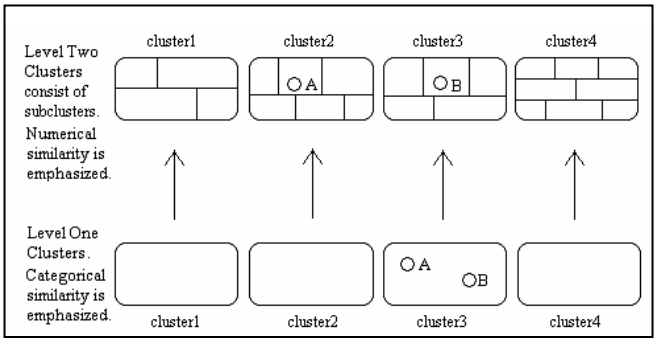


Figure 1. Overview of M-BILCOM clustering.

We emphasize that different types of data are used at levels one and two. At the first level the categorical data used represent something that has been observed to be true about the data set objects before the experiment takes place. For example the data at the first level might look as follows: Class:LIVE; SEX:male; STEROID:yes; FATIGUE:no; ANOREXIA:no . At the second level, on the other hand, the numerical data used represent the results of an experiment involving the data set objects. For example the data at the second level might look as follows: BILIRUBIN:0.39; ALBUMIN:2.1; PROTIME:10.

Our clustering method has the following requirements:

1. The coherence of each cluster should be maximized, considering both the numerical and categorical similarity of the objects.
2. Only the objects with highest categorical similarity to a cluster should form the basis for clustering at the first level.
3. The results of the first level clustering – which is the prior for the process - should not exert an overly strong effect on the second level, so that the second level clustering can escape a poor prior.
4. It should be possible to form a flexible number of clusters.
5. The similarity formula for comparing a mode to an object should increase an object's likelihood to be attached to a cluster as many CAs match the mode and as the CVs of those CAs increase.

We combined MULICsoft [2] and BILCOM [1] into an advanced clustering algorithm named M-BILCOM. This algorithm is similar to BILCOM [1], except that at the first level it also considers weights between 0.0 and 1.0 on the CAs - which we refer to as confidence values or CVs. This combined algorithm is especially useful in cases where the CAs have been perturbed and a CV between 0.0 and 1.0 has been assigned to each CA to indicate the certainty of correctness. Alternatively, M-BILCOM could be used on biological yeast data sets – such as SGD - where the certainty of correctness that exists on current knowledge is expressed as GO evidence codes [8, 12, 20].

3.1 First Level Clustering: MULICsoft

At the first level, clustering is performed using MULICsoft[1] which has a special similarity metric that incorporates CVs in the clustering process. MULICsoft is an extension of the k-Modes clustering algorithm for categorical data sets [18]. MULICsoft clusters only a subset of the data set objects. The number of clusters resulting from this level equals the final number of clusters desired, as illustrated in Figures 1 and 2.

The purpose of MULICsoft is to maximize the following similarity formula at each iteration, while ensuring that all objects may eventually be inserted in clusters:

$$similarity(object_n, mode_n)$$

where $object_n$ is the *nth* object in the data set to be clustered and $mode_n$ is the mode of the cluster to which $object_n$ is classified.

MULICsoft starts by reading all objects from the input file and storing them in a linked list *S*. The first object is inserted in a new cluster, the cluster's mode is set equal to the object's CAs and the object is removed from *S*. Then, it iterates over all objects that have not been classified in a cluster yet, to find the closest cluster. The closest cluster is determined for each unclassified object by comparing all clusters' modes with the object. The similarity between a mode and an object is determined using a special variation of the k-Modes similarity metric [2,18] that incorporates CVs in the clustering process and is described in Section 3.1.1.

The variable φ is maintained to indicate how strong the similarity has to be between an object and the closest cluster's mode for the object to be inserted in the cluster – initially φ equals 0, meaning that the similarity has to be very strong between an object and the closest cluster's mode. If the number of different CAs between the object and the closest cluster's mode are greater than φ, then, the object is inserted in a new cluster on its own and the cluster's mode is set equal to the object's CAs. If the number of different CAs between the object and the closest cluster's mode are less than or equal to φ, then, the object is inserted in the closest cluster and the mode is updated.

At the end of each iteration, all clusters with size one are removed, so their objects will be re-clustered at the next iteration. Thus, the clusters that persist are those containing at least two objects for which the required similarity can be found. Objects belonging to clusters with size greater than one are removed from the linked list of objects *S*, so those objects are not re-clustered.

At the end of each iteration, if no objects have been placed in clusters of size greater than one, then, the variable φ is incremented to represent how many CAs are allowed to differ next time. Thus, at the next iteration it will be more flexible and eventually more objects will be placed in clusters. Eventually, all objects will be given the opportunity to be placed in clusters, even if the closest cluster is not so similar. The iterative process may stop when all objects have been placed in clusters of size greater than one, or when φ is greater than a user-specified *threshold*.

The MULICsoft algorithm gives the opportunity for all objects to be eventually placed in clusters, because φ may continue increasing until all objects are classified. Even if, in the extreme case, an object with *m* CAs has only one or zero CAs similar to the mode of the closest cluster, it can be classified when $\varphi = m-1$ *or* $\varphi = m$, respectively.

Figure 2 illustrates the results of MULICsoft. Each cluster consists of many different "layers" of objects. The layer of an object represents how strong the object's similarity was to the mode of the cluster when the object was inserted. The cluster's layer in which an object is inserted depends on the value of φ. Thus, lower layers have a lower coherence – defined as the average similarity between all pairs of objects in that layer - and correspond to higher values of φ and to a more flexible similarity criterion for insertion. MULICsoft starts by inserting as many objects as possible at higher layers and then moves to lower layers, creating them as the need arises. If little similarity exists between an object and its closest cluster, the object will be inserted in a lower layer.

If an unclassified object has equal similarity to the modes of the two (or more) closest clusters, then the algorithm tries to resolve this 'tie' by comparing the object to the mode of the top layer of each of these clusters – the top layer of a cluster may be layer 0 or 1 or 2 and so on. Each cluster's top layer's mode was stored by MULICsoft when the cluster was created, so it does not need to be recomputed. If the object has equal similarity to the modes of the top layer of all of its closest clusters, the object is assigned to the cluster with the highest bottom layer. If all clusters have the same bottom layer then the object is assigned to the first cluster, since there is insufficient data for selecting the best cluster.

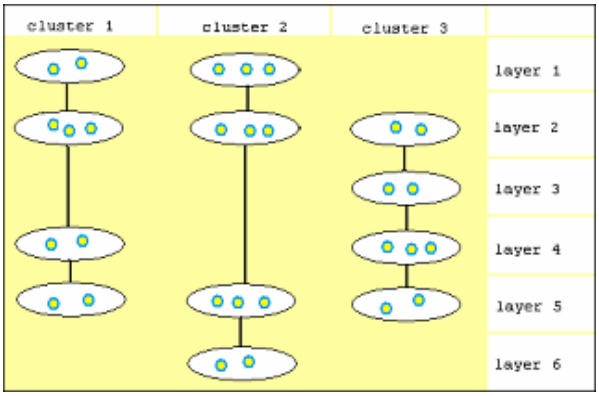


Figure 2. MULICsoft results. Each cluster consists of one or more different layers representing different similarities of the objects attached to the cluster.

The runtime complexity of MULICsoft is $O(N^2)$, where *N* represents the number of objects in the data set [2]. In most of our trials the runtime was less than 1 second.

The question remains of which objects to be clustered at the first level of M-BILCOM. The first level objects are those whose comparison to the mode of the closest cluster yields a result that is greater than or equal to a threshold *minimum_mode_similarity*. The rest of the objects are used at the second level, as described in Section 3.2. For this purpose, the user can specify a maximum value for φ - a value of *m-minimum_mode_similarity*, where *m* is the total number of categorical attributes in an object. When φ exceeds this value, any remaining objects are held for consideration instead in the second level. The reason we choose to insert in the first level clusters just the objects whose similarity to the closest mode yields a value higher than a threshold *minimum_mode_similarity* is because the objects that yield a low similarity to the closest mode are more likely to be inserted in the wrong cluster, as we show in [1,2]. Thus, the objects whose classification in clusters based on categorical similarity is not reliable enough, are clustered in the second level instead, where the numerical similarity of objects to second level clusters is more influential.

[1] http://www.cs.yorku.ca/~billa/MULIC/

3.1.1 The MULICsoft Similarity Metric

All CAs in an object have "weights" or CVs in the range 0.0 to 1.0 associated with them. We represent the *ith* weight of an object as w_i, the *ith* CA of object o as o_i and the *ith* value of mode μ as μ_i. The similarity metric used in MULICsoft for computing the similarity between a mode μ and an object o considers both the CAs and their weights. Our similarity metric amplifies the object positions having high weights, at pairs of CAs between an object o and a mode μ that have identical values.

$$similarity(o,\mu) = \sum_{i=1}^{m} \frac{6-(4\times w_i)}{5-(4\times w_i)} \times \sigma(o_i,\mu_i)$$

$$\text{where } \sigma(o_i,\mu_i) = \begin{cases} 1 & (o_i = \mu_i); \\ 0 & (o_i \neq \mu_i). \end{cases}$$

This similarity metric gives more importance to high weights (1.0) than low weights (0.1) at categorical attributes with identical values between the object and the mode.

Figure 3 shows that our similarity formula for comparing a mode to an object increases an object's likelihood to be attached to a cluster as many CAs match the cluster's mode and as the weights on those CAs increase. Each object in this example has 10 CAs and "weights" (or CVs). Figure 3 shows that an object will be much more likely to be assigned to a cluster if all CAs match the mode with high weights of 1.0, than if all CAs match the mode with medium weights of 0.5, than if all CAs match the mode with low weights of 0.1, than if 1 CA matches the mode with a high weight, than if 1 CA matches the mode with a low weight.

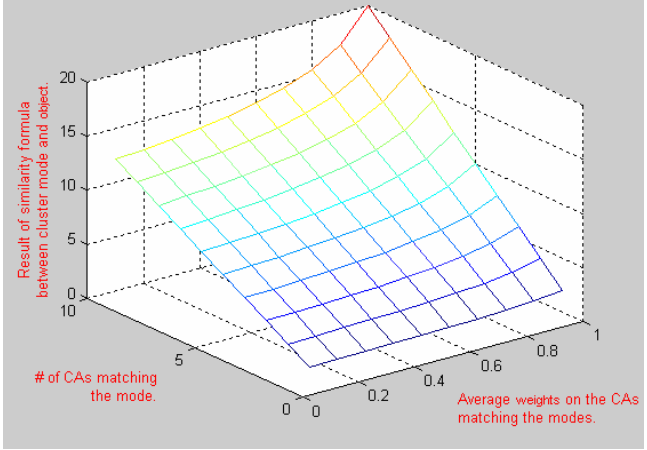


Figure 3. The function surface of *similarity*, using values for the weights between 0.0 and 1.0 as described previously.

3.1.2 Dealing with Outliers

MULICsoft can, eventually, put all the objects in clusters. When φ equals the number of attributes m, an unclassified object can be inserted in the lowest layer m of any existing cluster. This is undesirable if the object is an outlier and has little similarity with any cluster. The user can disallow this situation from happening, by specifying a *threshold* for φ that is less than the number of CAs m, so, when this *threshold* is reached any remaining objects are not classified and are treated as outliers. As discussed in [2] for clustering of software systems, the overall quality of the results often improves by treating the lowest-layer objects as outliers.

3.2 Second Level Clustering: BILCOM

The first level result is the input to the second level. The second level clusters all of the data set objects, including the objects clustered at the first level. The second level uses numerical data type similarity and the first level result as a prior. The second level clustering consists of 5 steps, whose rationale is to simulate maximizing the numerator of the Bayesian equation, as discussed in Section 3.1. The second level result is the output of the BILCOM process.

Step 1. One object in each first level cluster is set as a *seed*, while all the rest of the objects in the cluster are set as *centers*. The seed is an object that is at the top layer of the cluster – ideally in layer 0. The reason we choose for seed a top layer object is that the most influential objects at the second level should be those that have the minimum average distance to all other objects in the first level cluster. The MULIC paper [2] showed that objects at the top layer have a smaller average distance to all other cluster objects than lower level objects do.

If the top layer of a cluster is layer 0 then we have no difficulty in choosing the seed since all objects have the same CAs. If the top layer of a cluster is not layer 0 and it contains more than one object, then we choose the seed by comparing all top layer objects to the cluster's mode to find the closest object. If this does not resolve the ambiguity then we compare all top layer objects to the cluster's top layer mode – which was stored by MULIC when the cluster was created - to find the closest object. If all top layer objects have the same similarities to modes then we assign the seed to be the first top layer object, since there is insufficient information for choosing the best seed.

Step 2. Each *seed* and *center* is inserted in a new *second level subcluster*. The output of this step is a set of *subclusters*, referred to as *seed-containing* or *center-containing* subclusters, whose number equals the number of objects clustered at the first level.

Step 3. Each object that did not participate at the first level is inserted into the second level subcluster containing the most numerically similar *seed* or *center*. Numerical similarity for Steps 3-5 is determined by the *Pearson correlation coefficient* or the *Shrinkage-based similarity metric* introduced by Cherepinsky et al [9].

Step 4. Each center-containing subcluster is merged with its most numerically similar seed-containing subcluster. The most numerically similar seed-containing subcluster is found using our version of the ROCK goodness measure [14] that is evaluated between the center-containing subcluster in question and all seed-containing subclusters:

$$G(Ci,Cj) = \frac{link[Ci,Cj]}{size(Ci)\times size(Cj)}$$

$link[C_i,C_j]$ stores the number of cross links between subclusters C_i and C_j, by evaluating $\Sigma(o_q \in C_i,\ o_r \in C_j)\ link(o_q,o_r)$. $link(o_q,o_r)$ is a boolean value specifying whether a link exists between objects o_q and o_r . A link is set between two objects if the objects' numerical similarity is higher than a value *minimum_numerical_similarity*. The rationale for using a variation of ROCK's goodness measure for this step is that the link-based approach of ROCK adopts a global approach to the clustering problem, by capturing the global information about neighboring objects between clusters. It has been shown to be more robust than methods that adopt a local approach to clustering, like hierarchical clustering [14].

Step 5. The loop below refines the step 4 subcluster merges. All variables take real values in the range 0.0-1.0.

```
repeat {
    foreach (center-containing_subcluster)
        if
(numerical_similarity_of_center_subcluster_to_1st_
level_seed_cluster ×
categorical_similarity_of_center_to_seed_of_1st_le
vel_cluster >
numerical_similarity_of_center_subcluster_to_its_n
umerically_similar_2nd_level_cluster ×
categorical_similarity_of_center_to_seed_of_its_nu
merically_similar_2nd_level_cluster)
            merge center-containing_subcluster
to seed-containing_subcluster from 1st_level;
} until (no center-containing_subcluster changes);
```

The variable:

```
categorical_similarity_of_center_to_seed_of_1st_le
                    vel_cluster
```

represents the *categorical* similarity of the center *c* of a subcluster *C* to the seed *s*, such that *c* and *s* were in the same first level cluster.

The variable:

```
categorical_similarity_of_center_to_seed_of_its_nu
       merically_similar_2nd_level_cluster
```

represents the *categorical* similarity of the center *c* of a subcluster *C* to the seed of C's most numerically similar seed-containing subcluster *N* determined in step 4. The categorical similarity is computed as follows, where wc_i is the *ith* weight of the center and ws_i is the *ith* weight of the seed:

$$\text{similarity}(\text{center},\text{seed}) = \frac{\sum_{i=1}^{m} wc_i \times ws_i \times \sigma(\text{center}_i, \text{seed}_i)}{m}$$

where $\sigma(\text{center}_i,\text{seed}_i)= 1$ if $\text{center}_i=\text{seed}_i$, 0 otherwise.

The variables:

```
numerical_similarity_of_center_subcluster_to_1st_l
                evel_seed_cluster
numerical_similarity_of_center_subcluster_to_its_n
       umerically_similar_2nd_level_cluster
```

represent the *numerical* similarity of a subcluster *C* containing center *c* to the cluster containing seed *s*, such that *c* and *s* were in the same first level cluster, and to the cluster containing C's most numerically similar seed-containing subcluster *N* determined in step 4, respectively. These similarities include the subclusters that were merged to the clusters in previous iterations of the loop.

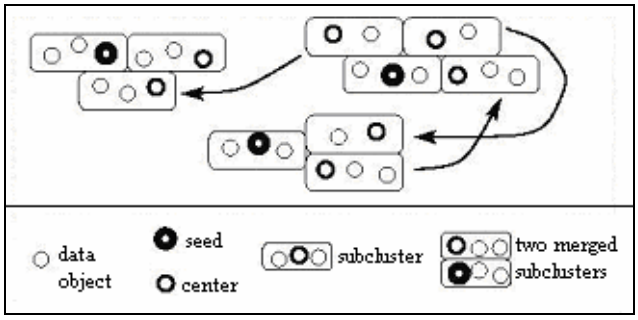


Figure 4. Steps 4 and 5 of second level of BILCOM clustering.

According to this loop, a subcluster *C* containing *center c* is attracted to the subcluster *S* containing *seed s*, such that *c* and *s* were in the same first level cluster. The attraction is stronger if there is high categorical similarity between *c* and *s* and lower if there is low categorical similarity between *c* and *s*. The subclusters *C* and *S* get merged if both the categorical similarity between *c* and *s* and numerical similarity between *C* and *S* are high enough. If *c* is not categorically similar enough to *s*, then, *C* should be likely to remain merged with its most numerically similar seed-containing subcluster *N* determined in step 4. Figure 4 shows steps 4 and 5.

We have done tests to show that BILCOM is able to *escape a poor prior* – for instance, if a center *c* was inserted in a first level cluster with weak similarity to the cluster mode, or if the similarity to the mode was erroneously high enough, or if *c* had wrong CAs with low confidence to be correct. The categorical similarity between the center *c* and the seed *s*, such that *c* and *s* were in the same first level cluster, is likely to return a low value when the prior is poor. In this case, the subcluster *C* containing center *c* will be likely to remain merged with its most numerically similar seed-containing subcluster *N* determined in step 4, instead of the subcluster *S* containing the seed *s*. Thus, the prior can be escaped and the data can be clustered correctly. In this case, *C* will not be merged to *S*, unless their numerical similarity is very high.

On the other hand, if the subcluster *C* containing center *c* is merged to the subcluster *S* containing the seed *s*, such that *c* and *s* were in the same first level cluster, then *C* must be numerically similar enough to *S*. This way we ensure that if a subcluster *C* is merged to the subcluster *S* that is suggested by the results of the first level clustering, the numerical similarity between *C* and *S* is high enough to support the merging.

The reason why the inequality comparison in step 5 considers the seeds of clusters, instead of the cluster modes, is that by considering similarity to seeds we are effectively giving the objects a *second chance* to reorganize and to escape their first level clustering, if the first level clustering was weak. Since the first level clustering was based on comparisons to modes that often yield wrong results and, therefore, objects may be attached to wrong clusters, the comparison in step 5 allows the similarities to be reconsidered. We showed in [2] that objects in the top layers 0 and 1 such as seeds have a higher average similarity to all other cluster objects than do lower layer objects.

4. TWO METHODS FOR IDENTIFYING SIGNIFICANT CAs IN A CLUSTER

This section first describes the real yeast data on which we applied the M-BILCOM clustering algorithm. Then, it presents two significance metrics (SMs) for determining the significance of a CA in a cluster and for supporting gene functional prediction.

4.1 Description of Real Yeast Data Sets

This algorithm is designed with the goal of applying it to numerical data sets for which some CAs exist and the confidence that the CAs are correct varies. We used numerical data derived from gene expression studies on the yeast *Saccharomyces cerevisiae*. These data sets were produced at Stanford to study the yeast cell cycle across time and under various experimental conditions and are available from the SGD database [9, 23]. When clustering this data set, we consider each gene to be a 'data object'. The data set contained 6,200 objects.

We represented *CAs* on a gene in terms of Gene Ontology (GO) and GOSlim annotations. GO is a dynamically controlled vocabulary that can be applied to many organisms, even as knowledge of gene and protein roles in cells is changing. GO is

organized along the categories of molecular function, biological process and cellular location [12]. GOSlim are GO annotations that represent high level knowledge on genes and are also organized along the categories of function, process and location. Most of the GO and GOSlim annotations on the yeast genes exist in the publicly accessible SGD database, along with GO evidence codes [8, 12, 20]. We created six pools of CAs for each gene and each pool contained GO annotations of a specific type. Three pools contained GO annotations for molecular function, biological process and cellular location of a gene. The other three pools contained GOSlim annotations for each GO annotation.

We attached CVs to the CAs to represent the confidence that the corresponding CA is correct. CVs are real numbers between 0.0 and 1.0, assigned to the CAs of a gene. Besides indicating the confidence that a CA is correct, the CVs on a gene also specify how strongly the gene's CAs should influence the clustering process. The CVs are also used in the significance metrics that we define below. We determined the CVs by using GO evidence codes. GO evidence codes symbolize the evidence that exists for any particular GO or GOSlim annotation [12].

GO evidence codes can be thought of in a loose hierarchy from strong evidence to weak evidence. For example, 'TAS' means 'Traceable Author Statement', while 'NAS' means 'Non-traceable Author Statement' [9]. We assigned a numerical CV to each of the GO evidence codes based on its location in the hierarchy, as shown in Figure 5. Section 7 discusses our justifications. NR and ND are set to 0.0, because they are used for annotations of 'unknown', so the CAs should not have an effect on the clustering process. In certain DBs (Swiss-prot-human) only 3 of these evidence codes are commonly used and the most commonly used one is TAS, which is at the top of the hierarchy, meaning that strong evidence exists [20]. We combined the CAs, CVs and gene expression data using Perl.

TAS - 1.0
IDA - 1.0
IMP - 0.8
IGI - 0.8
IPI - 0.8
ISS - 0.5
IEP - 0.5
NAS - 0.2
IEA - 0.1
ND - 0.0
NR - 0.0

Figure 5. GO Evidence Codes are mapped to CVs.

A CV primarily depends on whether the CA refers to something that has been *observed* to be true, as opposed to something that is just *believed* to be true. For example, a CA of a "cancerous tissue" refers to an observed phenomenon with a high CV, while a CA of a "non-cancerous tissue" refers to something that is just believed to be true, as the tissue might turn out later to be cancerous.

4.2 First SM: M-values that Consider P1-values and CVs are Assigned to Cluster CAs

Given a resulting cluster, we assigned a P1-value to each CA in the cluster; the term 'P1-value' was derived from the statistical 'P-value'. A P1-value measures whether a cluster contains a CA of a particular type more frequently than would be expected by chance [25]. A P1-value close to 0.0 indicates a frequent occurrence of the CA in the cluster, while a P1-value close to 1.0 its seldom occurrence. We multiplied the resulting P1-value with the reciprocal of the average of all CVs assigned to the CA in the cluster, 1/avg(CV), thus resulting in what we call an *M-value*. M-values allow us to take into consideration the probability that a particular CA occurs in the cluster more frequently than expected by chance, in addition to our confidence that the CA is correct in the cluster. For CAs that occur only once or twice in a cluster, a high P1-value results with an avg(CV) trivial to estimate.

4.3 Second SM: The Significance of a Second Level Subcluster's Classification in a Cluster

This significance metric was inspired by the loop of step 5 that refines the subclusters composing a larger second level cluster, as shown in Section 3.2. Specifically, each subcluster was assigned a significance number by evaluating a formula that considers both categorical (*CAsimilarity)* and numerical (*NAsimilarity)* similarity of the subcluster to the larger second level cluster:

*(weight1*CAsimilarity) +(weight2*NAsimilarity)*

The CAsimilarity for a subcluster is computed by evaluating a categorical variation of ROCK's goodness measure [16] between the subcluster and its larger cluster and multiplying the result by the percentage of genes in the subcluster that were assigned to it on the basis of categorical similarity (see Section 3.2 step 3). The NAsimilarity for a subcluster is computed similarly, by evaluating a numerical variation of ROCK's goodness measure [16] between the subcluster and its larger cluster and multiplying the result by the percentage of genes in the subcluster that were assigned to it on the basis of numerical similarity (see step 3). We set *weight2* in our trials to be higher than *weight1*, to ensure proper consideration of the numerical similarity of a subcluster.

The subclusters in an overall second level cluster for which the above metric yields the highest values are used for functional prediction by identifying and extracting the most significant genes' CAs in the cluster with highest avg(CV)s.

When a subcluster is placed in a larger second level cluster on the basis of high CA similarity (0.5-1.0) – regardless of whether it was assigned there at the beginning of the clustering process or joined it later - this is a factor that increases the significance. The NAsimilarity on the subcluster might be either high or low:

- high (0.5-1.0) in which case the significance of its membership is increased, because both CAs and NAs support the gene's classification in the cluster.
- low (0.1-0.4) in which case the significance of its membership is decreased, because CAs support the gene's classification in the cluster but NAs do not.

When a subcluster is placed in a larger second level cluster on the basis of low CA similarity (0.0-0.4) – regardless of whether it was assigned there at the beginning of the clustering process or joined it later - this is a factor that decreases the significance. The NAsimilarity on the subcluster is:

- always high (0.7-1.0). However, since the CA similarity is low the significance of its membership is decreased because NAs support the gene's classification in the cluster but CAs do not.

4.4 Functional Prediction for Uncharacterized Genes

Both of the above significance metrics (SM) were used for functional prediction of genes. Section 7 discusses our tests.

The M-values were used for functional prediction by taking for each cluster the CAs with the lowest M-values for molecular function, biological process, cellular location and for the GOSlim terms. Then, we applied these CAs to genes in the cluster having CAs labeled as 'Unknown'. Therefore, the CAs with the lowest M-values in a cluster were used to predict cellular roles of genes.

The second SM was used for functional prediction by identifying the subcluster with the highest significance in a larger second level cluster and identifying its genes' CAs with highest avg(CV)s in the cluster, as these were the most significant ones. Then, we applied the extracted CAs to other genes in the cluster having CAs labeled as 'Unknown'. Therefore, the CAs belonging

to the subcluster that had the highest significance were used to predict cellular roles of genes.

5. EXPERIMENTS ON YEAST DATA

We have validated M-BILCOM on mixed numerical and categorical yeast data. We used the yeast data sets shown in Table 1, having mixed categorical and numerical attribute values [7, 9]. However, we perturbed the CAs randomly and assigned statistical confidence values based on the probability that an attribute was perturbed or not. Tests and results are described below.

We represented CAs on a gene in terms of Gene Ontology (GO) - see Section 4. CAs were perturbed and the attribute values in the data set were assigned CVs between 0.1 and 1.0. For this purpose, for each CA we generated a *limit* in a range from 0.1 to 1.0 and, then, generated a random number ρ from 0.0 to 1.0. If ρ exceeded the *limit*, then we perturbed the CA by assigning it a value taken randomly from the set of possible values for that CA. The CV for the CA was set equal to the *limit* regardless of whether it was actually perturbed or not. This simulates the uncertainty that exists on current knowledge and that is expressed in SGD as GO evidence codes [8, 12, 20]. All attribute values on all objects were assigned a CV between 0.1 and 1.0 and objects whose CAs had lower CVs were more likely to have been perturbed than objects whose attribute values had higher CVs.

The yeast microorganism performs a constant cell-cycle. The yeast cell-cycle gene expression program is regulated by the nine known cell-cycle transcriptional activators, that control the flow from one stage of the cell-cycle to the next [7]. This serial regulation of transcriptional activators together with various functional properties suggests a way of partitioning cell-cycle genes into nine clusters, each one characterized by a group of transcriptional activators working together and their functions [7]. Table 1 shows our hypothesis about how the genes should be correctly grouped by transcriptional activators and cell-cycle functions. For instance, group 2 is characterized by the activators Swi6 and Mbp1 and the function involving DNA replication and repair at the juncture of G1 and S stages.

Table 1. Genes in the data set of Cherepinsky et al. [7] grouped by functions. This is our hypothesis about the correct clustering results.

Group	Activators	Genes	Functions
1	Swi4, Swi6	CLN1, CLN2, GIC1, MSB2, RSR1, BUD9, MNN1, OCH1, EXG1, KRE6, CWP1	Budding
2	Swi6, Mbp1	CLB5, CLB6, RNR1, RAD27, CDC21, DUN1, RAD51, CDC45, MCM2	DNA replication and repair
3	Swi4, Swi6	HTB1, HTB2, HTA1, HTA2, HTA3, HHO1	Chromatin
4	Fkh1	HHF1, HHT1, TEL2, ARP7	Chromatin
5	Fkh1	TEM1	Mitosis control
6	Ndd1, Fkh2, Mcm1	CLB2, ACE2, SWI5, CDC20	Mitosis control
7	Ace2, Swi5	CTS1, EGT2	Cytokinesis
8	Mcm1	MCM3, MCM6, CDC6, CDC46	Prereplication complex formation
9	Mcm1	STE2, FAR1	Mating

Cherepinsky et al. [7] defined a notation to represent the resulting cluster sets and a scoring function to aid in their comparison. Each cluster set is written as:

$$\{x \rightarrow \{\{y_1, z_1\}, \{y_2, z_2\}, \ldots, \{y_{n_x}, z_{n_x}\}\}\}_{x=1}^{\#\text{ of groups}},$$

where x denotes the group number as described in Table 1, n_x is the number of clusters the members of group x appear in, and for each cluster $j \in \{1,..., n_x\}$ there are y_j genes from group x and z_j genes from other groups in Table 1. A value of * for z_j denotes that cluster j contains additional genes, although none of them are cell-cycle genes. The cluster set can then be scored as follows:

$$\mathrm{FP}(\gamma) = \frac{1}{2}\sum_{x}\sum_{j=1}^{n_x} y_j \cdot z_j$$

$$\mathrm{FN}(\gamma) = \sum_{x}\sum_{1 \le j < k \le n_x} y_j \cdot y_k$$

$$\text{Error score}(\gamma) = \mathrm{FP}(\gamma) + \mathrm{FN}(\gamma).$$

We have compared the error rates of M-BILCOM to those of AutoClass [24] and BILCOM [1] applied to the "perturbed" yeast data set. First, we applied AutoClass [24] to the mixed categorical and numerical yeast data set. Table 2 shows the results.

Table 2. Clustering results of AutoClass.

Cluster	Genes
1	CLN1, CLN2, GIC1, GIC2, MSB2, RSR1, BUD9, MNN1, OCH1, EXG1, KRE6, CWP1, CLB5, CLB6, RAD51, CDC45, HTB1, HTA2, HHO1, TEL2
2	ARP7, TEM1, CLB2, ACE2, SWI5, CDC20, CTS1, EGT2, MCM3, MCM6, CDC6, CDC46, STE2
3	RNR1, RAD27, CDC21, DUN1, MCM2, HTB2, HTA1, HHF1, HHT1, FAR1

Given the hypothesis in Table 1 and the set of AutoClass results shown in Table 2, the resulting clusters with the error score can be written as follows, according to the error rate introduced by Cherepinsky et al. [7]:

{1->{{11,9}}, *2->{{4,16},{5,5}},* *3->{{3,17},{2,8}},* *4->{{1,19},{1,12},{2,8}},* *5->{{1,12}},* *6->{{4,9}},* *7->{{2,11}},* *8->{{4,9}},* *9->{{1,12},{1,9}} }.*	*FP = 265* *FN = 32* *Error = 297*

We have also applied BILCOM [1] to the mixed categorical and numerical yeast data set. Table 3 shows the results.

Table 3. Clustering results of BILCOM using as numerical similarity metric between objects the Pearson Correlation Coefficient [7] and a max value for φ of 7.

Cluster	Genes
1	RSR1, HHT1, ARP7, BUD9, CTS1
2	KRE6, CWP1
3	RNR1, CDC45, MCM3, CDC46, MCM2
4	EXG1, EGT2
5	MCM6, CDC6
6	HHF1, HTB2, HTA2
7	HTB1, HTA1, HHO1

8	GIC1, TEL2, GIC2, MSB2
9	FAR1, STE2, ACE2, SWI5, TEM1
10	RAD27, CDC21, DUN1
11	CLN2, RAD51, MNN1, CLN1, CLB6, OCH1, CLB5, CLB2, CDC20

Given the hypothesis in Table 1 and the set of BILCOM results shown in Table 3, the resulting clusters with the error score can be written as follows, according to the error rate introduced by Cherepinsky et al. [7]:

{1->{{4,4},{1,3},{1,1},{2,1},{2,0}}, *2->{{3,0},{3,2},{3,5}},* *3->{{3,0},{2,1}},* *4->{{2,3},{1,2},{1,4}},* *5->{{1,4}},* *6->{{2,3},{2,7}},* *7->{{1,4},{1,1}},* *8->{{2,0},{2,3}},* *9->{{2,3}} }.*	*FP = 49* *FN = 47+13+24* *Error = 133*

We have also applied M-BILCOM to the mixed categorical and numerical yeast data set. Table 4 shows the results.

Table 4. Clustering results of M-BILCOM using as numerical similarity metric between 2 objects the Pearson Correlation Coefficient [7] and a max value for φ of 7.

Cluster	Genes
1	RSR1, BUD9, CTS1
2	KRE6, ARP7, HHT1, CWP1
3	RNR1, CDC45, MCM3, STE2, CDC46, MCM2
4	EXG1, EGT2
5	MCM6, TEM1, CDC6
6	HHF1, HTB2, HTA2
7	HTB1, HHO1, HTA1
8	GIC1, TEL2, GIC2, MSB2
9	FAR1, ACE2, CDC20, SWI5
10	RAD27, CDC21, MNN1, DUN1, RAD51
11	CLN2, CLN1, CLB6, CLB5, OCH1, CLB2

Given the hypothesis in Table 1 and the set of M-BILCOM results shown in Table 4, the resulting clusters with the error score can be written as follows, according to the error rate introduced by Cherepinsky et al. [7]:

{1->{{3,3},{2,2},{2,1},{1,1},{2,2},{1,4}}, *2->{{4,1},{3,3}},* *3->{{3,0},{3,0}},* *4->{{2,2},{1,2},{1,3}},* *5->{{1,2}},* *6->{{3,1},{1,5}},* *7->{{1,2},{1,1}},* *8->{{2,4},{2,1}},* *9->{{1,5},{1,3}} }.*	*FP = 38* *FN = 35+49* *Error = 122*

The error rates are summarized in table 5 below. The M-BILCOM error rate is lower than BILCOM and AutoClass.

Table 5. Comparative error rates of algorithms applied to the "perturbed" yeast data set.

Clustering Tool	Cherepinsky Error rate
M-BILCOM	122
BILCOM	133
AutoClass	297

6. EXPERIMENTS ON SIMULATED DATA

We generated artificial data sets that simulate the results by Spellman et al [23] who showed that in each cluster there is a consistent pattern of numerical attribute values (NAs) that appear frequently and that different CAs are characteristic of different clusters. We used numerical data derived from gene expression studies on the yeast *Saccharomyces cerevisiae*. These data sets were produced at Stanford to study the yeast cell cycle across time and under various experimental conditions and are available from the SGD database [9, 23]. The data set contained 6,200 genes. The purpose of our simulation was to assign CAs to each gene based on the numerical gene expression data [9], in such a manner that the assignment of CAs simulates knowledge about the role of genes in the yeast cell cycle.

We assigned 6 CAs on each gene based on the NAs, representing the genes' action during cell cycle. The assignment of the CAs followed a pattern that simulates existing knowledge on the role of genes in the yeast cell cycle. The assigned CAs split the objects into a number of well-defined groups, which we attempt to retrieve using clustering; thus, a different set of attribute values had to be used for each group. This simulates the results by Spellman et al, who showed that in each cluster there is a consistent pattern of NAs that appear frequently and that different CAs are characteristic of different clusters [23]. We assigned the 6 CAs using the following strategy: (1) The first CA split the genes into cell cycle phases and has the values G1, S, G2, M, M/G1, or unknown. This CA was set for each gene based on the experimental point at which the gene reaches its peak expression level, indicating what cell cycle phase it is likely to be involved in. This simulates the cell cycle phases as they are derived by Spellman et al and the genes that are likely to be involved in each phase. (2)(3) The second and third CAs were set for each gene based on whether the gene's expression level peaks at the phase of the cell cycle at which it is active. The second CA, which can take 6 values (A,B,C,D,E,unknown), simulates an overall process like *DNA replication* or *transport of essential minerals and organic compounds across the cell membrane*. The third CA, which can take 11 values (F,G,H,I,J,K,L,M,N,O,unknown), simulates a more specific function like *DNA polymerases* or *nucleotide synthesis* or *initiation of DNA synthesis*. The purpose of the second and third CAs was to further subdivide the groups created by the first CA into subgroups. The first three CAs are influential enough to classify each gene in a cluster. Thus, the clusters that we will retrieve are based primarily on the first three CAs. (4) The fourth CA was set for each gene based on whether the gene is observed to reach its peak expression level right before the G1 stage. This CA simulates the *start of mitosis* and can take 3 values (high,low,unknown). (5) The fifth CA was set for each gene based on whether the gene is observed to reach its peak expression level at the end of the cell cycle. This CA simulates the *exit from mitosis* and can take 3 values (high,low,unknown). (6) Finally, the sixth CA was set for each gene based on whether the gene is observed to reach its peak expression level right before the S stage. This CA can take 3 values (high,low,unknown). The fourth to sixth CAs on their own may or may not classify each gene into a clear group.

Furthermore, we perturbed the CAs to simulate noise in the resulting data set. Our aim was to use M-BILCOM to retrieve the known underlying cluster structure effectively. A significant outcome of our experiments was to show that given the genes whose CAs were not perturbed in the simulation (most of which

are likely to have high CVs) a fair number of genes were assigned to the correct clusters to which they were categorically similar and were not assigned to the incorrect clusters to which they might be numerically similar. The basis for this is that most of these genes had a high confidence overall. Another significant outcome of our experiments was to show that given the genes whose CAs were perturbed in the simulation (most of which are likely to have low CVs) a fair number of genes were assigned to the correct clusters to which they were likely to be numerically similar and were not assigned to the incorrect clusters to which they were categorically similar. The basis for this is that most of these genes had a low confidence overall.

Many CAs were perturbed and the attribute values in the data set were assigned CVs between 0.1 and 1.0. For this purpose, for each CA we generated a *limit* in a range from *0.4* to 1.0 and then, generated a random number ρ from 0.0 to 1.0. If ρ exceeded the limit, then we perturbed the CA by assigning it a value taken randomly from the set of possible values for that attribute. The CV for the CA was set equal to the *limit,* regardless of whether it was actually perturbed or not. This simulates the uncertainty that exists on current knowledge and that is expressed in SGD as GO evidence codes [8, 12, 20]. In the produced data set 2,024 data objects had their original attribute values modified out of 6100. All CAs on all data objects were assigned a CV between 0.4 and 1.0 and objects with lower CVs were more likely to have been modified than objects with higher CVs.

We clustered the simulated data set into 20 clusters. This number of clusters was derived from the number of combinations of values that the first three CAs of each object can take in our simulated data and because this number of clusters allowed the algorithm to converge in a reasonable amount of time. Table 6 shows the statistics for all 20 clusters, though we ignore the results for clusters whose size was too small.

What is most noteworthy in Table 6 are clusters 8, 11, 15, because all of their data objects had their CAs modified during our simulation (see column 4). As can be seen, many of the data objects in these clusters had original values for their first 3 CAs consistent with the most representative CAs {A,B,C} for the cluster (see columns 5,6). Furthermore, *all* of the objects with original values for their first 3 CAs equal to the most representative CAs {A,B,C} for the cluster, were objects whose CAs had been modified during the simulation to different values (see column 7).

Table 6. Results for clustering the data set into 20 clusters. We do not show results for clusters whose size was too small.
1 – Cluster #
2 - Number of objects in the cluster
3 - Most common values {A,B,C} on the objects' first 3 CAs
4 - Ratio *X* of objects in the cluster that had CAs modified during the simulation
5 - Ratio of *X* that had an original CA very close to {A,B,C}
6 - Ratio *P* of objects in the cluster that had an original CA very close to {A,B,C}
7 - Ratio of *P* that had its CAs modified during the simulation
8 - Number of merged second level subclusters

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	2032	{M,D,L}	616/2032	217/616	1047/2032	217/1047	537
2	1186	{MG1,E,N}	305/1186	202/305	1102/1186	202/1102	180
3	724	{G2,C,J}	177/724	111/177	672/724	111/672	121
4	317	{G1,A,F}	83/317	48/83	302/317	48/302	44
5	709	{S,B,H}	94/709	24/94	684/709	24/684	183
6	218	{M,D,M}	50/218	26/50	198/218	26/198	59
8	66	{MG1,E,N}	66/66	22/66	22/ 66	22/22	9
11	71	{MG1,E,N}	71/71	27/71	27/ 71	27/27	3
15	74	{MG1,E,N}	74/74	24/74	24/ 74	24/24	2
20	333	{S,B,H} and {S,B,I}	180/333	148/180	260/333	148/260	13

Another interesting result is cluster 2, in which the most prominent genes are those with the values {MG1, E, N} for their first 3 CAs. 202/1102 objects had their CAs modified to a totally different value, but were, nevertheless, assigned to the correct cluster because they had low CVs (see column 7). This shows that our algorithm can overcome a poor prior that is likely to be incorrect and can still produce correct results by using numerical clustering instead. In this cluster, from all objects assigned to it that had their CAs modified (305/1186, as shown in column 4) 202/305 had CA values of {MG1, E, N} or {MG1, E, O} (see column 5). The total cluster size was 1186 and consisted of 180 merged second level subclusters (see column 8). Four of the merged clusters contained a vast majority of objects with modified CAs. All of these clusters had a substantial portion - or a majority - of objects with original CA values of {MG1, E, N}.

7. VALIDATION OF PREDICTIONS

7.1 First Significance Metric

Our strategy for validating the accuracy of the functional predictions is to reclassify certain genes' CAs as 'Unknown' before the clustering process and we aim to predict the correct genes' cellular roles using the cluster CAs pointed out by the SM. The CAs to be set to 'Unknown' were chosen to have a high average(CV) over all their occurrences in the cluster, because these are primarily the ones that we would like to be able to predict correctly. The process described next helps us to determine how likely genes are to be assigned their correct CAs.

We iterated over the genes in the cluster with CAs labeled as 'Unknown'. To assess the effectiveness of the technique, we verified that the original CAs of these genes correlated better to the cluster CAs with low M-values - that are pointed out by our SM - than those with high M-values. This correlation signified the likelihood that the genes' CAs labeled as 'Unknown' would be assigned their original values, by using CAs with low M-values that are pointed out by our SM. A relatively large number of genes' CAs labeled as 'Unknown' should be likely to be re-assigned their original values using CAs with low M-values in the cluster, because a low M-value indicates that a CA occurs frequently amongst the cluster's genes and that a CA is likely to be correct.

We initially clustered the yeast data into 5 clusters. Table 7 describes some CAs that were pointed out in all 5 clusters by the

SM, after the CAs with the highest average(CV) in each cluster were set to 'Unknown' and the set was clustered.

Table 7. CAs pointed out in 5 clusters as the most significant. The CAs pointed out in clusters 1-5 as having the lowest M-values - the most representative ones for the cluster - correlated with the CAs in the original cluster that were set to 'Unknown'.

Cluster	Some of the CAs pointed out in each cluster as having low M-values (meaning they occurred frequently and had high avg(CV)) after the CAs with the highest avg(CV) in each cluster were set to 'Unknown' and the set was clustered.
1	vacuolar membrane, ubiquitin-specific protease, small nuclear ribonucleoprotein complex, glycolysis, 3'-5' exoribonuclease, cytosolic small ribosomal subunit, lipid particle, cytosolic large ribosomal subunit, tricarboxylic acid cycle
2	rRNA modification, ATP dependent RNA helicase, nuclear pore, structural molecule, small nucleolar ribonucleoprotein complex, snoRNA binding, mediator complex
3	cytosol, proteasome endopeptidase, non-selective vesicle fusion, translation initiation factor
4	transcription initiation from Pol II promoter, general RNA polymerase II transcription factor, nucleus
5	endoplasmic reticulum membrane, component:endoplasmic reticulum

We have also performed these tests on the yeast data producing 35 and 71 clusters. We provide a concrete example of the utility of our technique for 35 clusters, by focusing on the second cluster having 224 genes. In the original clustering, the following CAs were pointed out as having the lowest M-values: function:transcription regulator, component:nucleus, process:transport, process:cell growth and/or maintenance, process:metabolism, function:transporter, nucleus(a specific, granular annotation). We focus on the 2 most significant (representative) CAs for the cluster:

1) *component:nucleus* occurred in 160 genes in this cluster and had an average(CV) of 1.0 across all genes. Some genes with this annotation were YOR064C, YBR247C, YDR205W, YDR206W, YFR023W, YKL117W, YPR196W, YOR141C, YOL116W, YOR294W, YDR076W, YFR037C, YNL148C, YDR510W, YLR074C, YPL049C, YDL064W, YML109W, YNL016W.
2) *nucleus(a specific, granular annotation)* occurred in 82 genes in this cluster and had an average(CV) of 0.904878 across all genes. Some genes with this annotation were YBR247C, YDR205W, YDR206W, YFR023W, YKL117W, YPR196W, YOL116W, YOR294W, YDR076W, YFR037C, YNL148C, YLR074C.

In different trials we set all CAs of many genes in which these two values occurred originally to 'Unknown' and then re-clustered the data set into 35 clusters. Using the results, we were able to predict correctly that these genes should be annotated as either component:nucleus or nucleus(a granular annotation) by extracting the CAs with lowest M-values. This means that the SM predicted these genes to have their original correct CAs, after setting them to 'Unknown', re-clustering the data set and extracting the CAs with lowest M-values.

Figure 6 illustrates the results obtained for a CA with value *A* (component:nucleus) that initially occurred in 160 objects in the cluster. As an increasing number of occurrences of value *A* in the cluster were set to 'Unknown' and the set was re-clustered, the same value *A* still qualified as one of the most significant values in the cluster and remained applicable to a relatively large number of objects (i.e. genes).

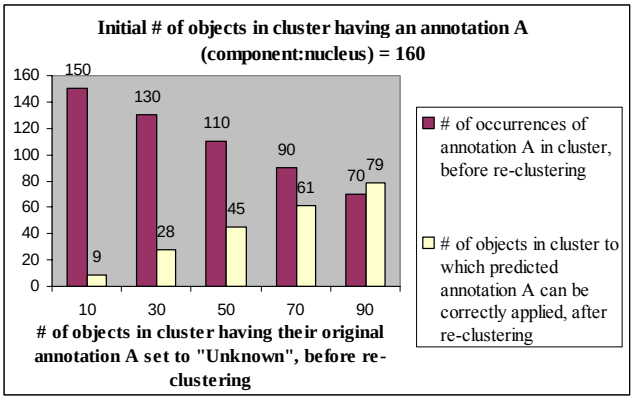


Figure 6. Results for different trials of setting occurrences of 'component:nucleus' to 'Unknown' and re-clustering the set.

7.2 Second Significance Metric

Our strategy for validating the accuracy of the functional predictions was to reclassify the CAs of certain genes as 'Unknown' before the clustering process and attempt to predict the correct genes' cellular roles using the cluster CAs pointed out by the SM. The CAs set to 'Unknown' were primarily ones with a high average(CV) over all their occurrences in the cluster, because these were primarily the ones that we would like to be able to predict correctly. The process described next helped us to determine how likely genes were to be assigned their correct CAs.

We iterated over the CAs in the cluster that were labeled as 'Unknown'. To assess the effectiveness of the technique, we verified that the original CAs of these genes correlated better to the cluster CAs pointed out as having the highest significance. CAs pointed out as highly significant were ones occurring frequently across the cluster's genes with high avg(CV). This correlation signified the likelihood that the genes' CAs labeled as 'Unknown' would be re-assigned their original values, by using CAs that were pointed out by the SM. A reasonable number of genes' CAs should be likely to be assigned their original values using the CAs pointed out by the SM.

We initially clustered the yeast data into 35 clusters, each of which contained a number of smaller subclusters. The second level subclusters pointed out by the SM as significant enough were those containing genes:

1) YHR053C (SM>1 ; 80% of genes not having CV high enough ; 23 genes total),
2) YDL179W (SM>1 ; 96% of genes not having CV high enough ; 104 genes total),
3) YKL182W (SM>0.58 ; 94% of genes not having CV high enough ; 210 genes total),
4) YKR075C (SM>0.44 ; 96% of genes not having CV high enough ; 27 genes total),
5) YLR342W (SM>0.10 ; 91% of genes not having CV high enough ; 22 genes total),
6) YMR246W (SM>0.88 ; 75% of genes not having CV high enough ; 61 genes total),
7) YJL079C (SM>0.06 ; 75% of genes not having CV high enough ; 4 genes total),
8) YCR005C (SM>0.58 ; 63% of genes not having CV high enough ; 470 genes total),

9) YMR186W (SM>0.06 ; 50% of genes not having CV high enough ; 8 genes total),
10) YBR029C (SM>1 ; 0% of genes not having CV high enough; 1 gene total),

The reason other subclusters yielded low significance was because a majority of their genes had high average(CV) over their CAs, so most genes were assigned on the basis of categorical similarity rather than on the basis of numerical similarity. Thus the dominant factor in the significance metric was low and the overall result was low.

We next needed to identify the CAs in these clusters with the highest average(CV) throughout the entire cluster. We identified the following CAs, for each of the subclusters listed above:

1) copper binding, avg(CV) 0.5 ; cytosol, avg(CV) 1.0
2) cell cycle, avg(CV) 0.5
3) fatty-acid synthase complex, avg(CV) 1.0 ; fatty acid biosynthesis, avg(CV) 1.0 ; vacuole (sensu Fungi), avg(CV) 0.8 ; vacuole inheritance, avg(CV) 0.8 ; thiol-disulfide exchange intermediate, avg(CV) 0.5 ; plasma membrane, avg(CV) 1.0 ; tricarboxylic acid cycle, avg(CV) 1.0
4) cytoplasm, avg(CV) 1.0
5) 1,3-beta-glucan synthase, avg(CV) 0.55
6) long-chain-fatty-acid-CoA-ligase, avg(CV) 0.55 ; lipid metabolism, avg(CV) 0.75 ; lipid particle, avg(CV) 1.0
7) nuclear membrane, avg(CV) 1.0
8) glyoxylate cycle, avg(CV) 1.0 ; peroxisomal matrix, avg(CV) 0.95 ; folic acid and derivative biosynthesis, avg(CV) 0.95 ; pantothenate biosynthesis, avg(CV) 0.8 ; allantoin catabolism, avg(CV) 0.8 ; purine nucleotide biosynthesis, avg(CV) 0.95 ; helicase, avg(CV) 0.5 ; spore wall assembly, avg(CV) 0.8 ; RAB-protein geranylgeranyltransferase, avg(CV) 0.55 ; protein amino acid geranylgeranylation, avg(CV) 1.0 ; RAB-protein geranylgeranyltransferase complex, avg(CV) 1.0
9) response to stress, avg(CV) 0.75
10) phosphatidate cytidylyltransferase, avg(CV) 1.0 ; phosphatidylserine metabolism, avg(CV) 1.0 ; mitochondrion, avg(CV) 1.0

In different trials we set the CAs of many genes in which these values originally occurred to 'Unknown' in each of these clusters and re-clustered the entire data set. The same values were still pointed out by the SM as highly significant in the corresponding clusters. This encouraged us to re-assign the original values to the genes whose CAs were set to 'Unknown', which we interpret as a success of our approach.

7.3 Assessment of Clustering Stability

The GO Evidence Codes (GOECs) form a loose hierarchy from strong evidence to weak evidence. The top GOECs in the hierarchy represented by 'TAS' and 'IDA' are mapped to a CV of 1.0 while the bottom GOECs 'ND' and 'NR' are mapped to 0.0. We want to show that all other GOECs falling between these extremes in the hierarchy are assigned a CV that allows objects to be partitioned the best way in the clustering process. We used the simulated yeast data set from Section 6 to determine how sensitive the final results are to changes in the spacing between the CVs.

We did a trial using GOECs from the top 3 hierarchy scales: TAS/IDI, IMP/IGI/IPI, ISS/IEP. We set a randomly chosen 1/3 of the CVs in the data set to TAS/IDI, 1/3 to IMP/IGI/IPI and 1/3 to ISS/IEP. Then we set all CVs of objects falling in 5 classes A-E in the data set to the middle GOEC of IMP/IGI/IPI. By mapping this set of GOECs to corresponding CVs of 1.0, 0.8, 0.5, the objects belonging in classes A-E were slightly better partitioned from other objects than when mapping to CVs of 1.0, 0.9, 0.6.

Then we repeated this trial by using GOECs from the bottom 3 hierarchy scales: ISS/IEP, NAS, IEA. By mapping this set of GOECs to corresponding CVs of 0.5, 0.2, 0.1, the objects belonging in classes A-E were slightly better partitioned from other objects than when mapping to CVs of 0.5, 0.3, 0.2.

We also assess the stability of the clustering to perturbations in the data, to determine the reproducibility of the results [26, 27]. Our 'perturbed' data includes changes in the spacing between the CVs and different perturbations of the simulated yeast data set. We then re-cluster the perturbed data and compute indices - such as R-index and D-index [27] - to determine how much the clustering has changed. R-index measures the proportion of pairs of objects within a cluster for which the members of the pair remain together in the perturbed re-clustered data [27]. D-index measures the number of omissions and additions comparing an original cluster to a best-matching cluster in the perturbed re-clustered data [27]. The R-index values were greater than 0.97 and the D-index values were less than 4.5, for several trials involving different mappings of GOECs to CVs and different perturbations of the simulated yeast data set. This indicates high reproducibility of the clustering results.

8. DISCUSSION: USING SIGNIFICANCE METRICS FOR DERIVING POTENTIAL GENE FUNCTIONS

Biologists will find this method useful for deriving hints about potential functions of genes or proteins. The hints that are derived as to a gene's function can later be validated experimentally. This will save time and money from the experimentalists' side. In our experiments with the yeast cell cycle data set, the utility of the significance metrics (SMs) is especially evident from the fact that the vast majority of genes in each cluster or subcluster analyzed had all CAs set to 'Unknown' meaning that no knowledge exists. For example, when analyzing the subcluster containing YHR053C using the second SM, only 6 out of 20 genes had some kind of CA, while the other 14 genes had CAs set to 'Unknown'. Our SM could point out the most representative CAs that are likely to be applicable to the other 14 genes and these functional hints can be tested experimentally.

M-values are useful for identifying the most representative CAs in clusters with a plethora of CAs that have *high CVs*. M-values allow one to identify the CAs in this pool that appear frequently (with a low P1-value) so as to apply them to other genes. In our experiments with the yeast cell cycle data set we realised that although 1185 second level subclusters had been produced in total, most of those (1175 = 1185-10 subclusters) had a majority of genes with an average CV over their CAs that was considered high. These genes were assigned to the clusters on the basis of categorical rather than numerical similarity, according to step 3. M-values could be utilized on these 1175 subclusters, or could be utilized on the overall clusters produced as an end result by the algorithm (the total number of clusters was 5, 35 and 71).

The *second* SM applies primarily for identifying the most representative CAs in clusters with a plethora of CAs that have *low CVs*. The second SM allows us to identify the few CAs that have high CVs in these clusters, so as to apply them to other genes. In our experiments, only 10 second level subclusters out of 1185 in total, had a majority of genes with an average CV across their CAs that was considered low enough. These genes were

assigned to the clusters on the basis of numerical rather than categorical similarity, according to step 3. The second SM could be utilized on these 10 subclusters.

Future work will include justifying on a theoretical basis the mapping of GO Evidence Codes to CVs. We will be applying this algorithm to more gene expression data sets for organisms on which low quality CAs exist. Furthermore, we will be developing more significance metrics for the M-BILCOM clustering results.

9. CONCLUSION

When clustering low quality data with uncertainties about the data's correctness, we need to develop our ability to integrate data from various sources, including numerical data and categorical data. Furthermore, we need to be able to claim that what we see in a clustering analysis is more reliable or less reliable and, therefore, may or may not be a strong basis for making decisions. In this paper we have described the novel M-BILCOM clustering algorithm for mixed numerical and uncertain categorical data sets that incorporates CAs and CVs representing certainty about the correctness of the CAs. This clustering algorithm inspired us to define two new significance metrics for extracting from each cluster the most significant CAs, that form a strong basis for deriving conclusions about the CAs of other objects in the cluster. We showed that these significance metrics can be successfully used for finding the most significant CAs in a cluster. For genomic data sets we applied the significant CAs to other genes in the cluster for functional prediction. We experimented with this clustering tool on highly noisy yeast data sets for which the correct results were known. We showed that M-BILCOM can reliably identify the cluster structure in simulated data sets.

10. REFERENCES

[1] Andreopoulos, B., An, A. and Wang, X. (2005) BILCOM: Bi-level Clustering of Mixed Categorical and Numerical Biological Data. Technical report CS-2005-01. York University. http://www.cs.yorku.ca/techreports/

[2] Andreopoulos, B., An, A. and Wang, X. (2004) MULIC: Multi-Layer Increasing Coherence Clustering of Categorical Data Sets. Technical report CS-2004-07. York University.

[3] Andreopoulos, B., An, A. and Wang, X. (2003) Significance Metrics for Clusters of Mixed Numerical and Categorical Yeast Data. Technical report CS-2003-12. York University.

[4] Adryan B. and Schuh R. (2004) Gene ontology-based clustering of gene expression data, Bioinformatics, Nov 2004; 20: 2851 - 2852.

[5] Ben-Dor A., Shamir R., Yakhini Z. (1999) Clustering Gene Expression Patterns. Journal of Computational Biology 6(3/4): 281-297.

[6] Brown M.P.S. Grundy W.N., Lin D., Cristianini N., Sugnet C.W., Furey T.S., Manuel A., Haussler D. Knowledge-based analysis of microarray gene expression data by using support vector machines. PNAS 97(1), 262-267, 2000.

[7] Cherepinsky V., Feng J., Rejali M. and Mishra B. (2003) Shrinkage-Based Similarity Metric for Cluster Analysis of Microarray Data. PNAS 100(17): 9668-9673.

[8] Dwight SS, Harris MA, Dolinski K, Ball CA, Binkley G, Christie KR, Fisk DG, Issel-Tarver L, Schroeder M, Sherlock G, Sethuraman A, Weng S, Botstein D, Cherry JM. (2002) Saccharomyces Genome Database provides secondary gene annotation using the Gene Ontology. *Nucleic Acids Research* 30: 69-72.

[9] Eisen, M.B. & Brown, P.O. (1999) DNA arrays for analysis of gene expression. *Methods Enzymol.* 303, 179-205.

[10] Eisen MB, Spellman PT, Brown PO, Botstein D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci USA*. 95(25):14863-8, 1998.

[11] Fasulo D. (1999) An Analysis of Recent Work on Clustering Algorithms, Technical Report # 01-03-02, Department of Computer Science & Engineering, University of Washington.

[12] The Gene Ontology Consortium (2001). Creating the gene ontology resource: design and implementation. *Genome Research* 11: 1425-1433.

[13] Goebel, M. & Gruenwald, Le (1999). A survey of data mining and knowledge discovery software tools. *ACM SIGKDD Explorations* 1, 20-33 .

[14] Golub, T. R. et al. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 286, 531-537.

[15] Grambeier J., Rudolph A. (2002) Techniques of Cluster Algorithms in Data Mining. *Data Mining and Knowledge Discovery* 6: 303-360.

[16] Guha S., Rastogi R., Shim K. (2000). ROCK: A Robust Clustering Algorithm for Categorical Attributes. *Information Systems* 25(5): 345-366.

[17] Hartigan, J. A. (1975) Clustering algorithms. (John Wiley and Sons, New York, 1975).

[18] Huang Z. (1998) Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery* 2(3): 283-304.

[19] Huang, Z. (1997) Clustering Large Data Sets with Mixed Numeric and Categorical Values. *Knowledge discovery and data mining: techniques and applications*. World Scientific.

[20] Lord P.W., Stevens R.D., Brass A. and Goble C.A. (2003). Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. *Bioinformatics* 19: 1275-83.

[21] Pasquier C., Girardot F., Jevardat de Fombelle K., and Christen R. (2004) THEA: Ontology driven analysis of microarray data. Bioinformatics, Nov 2004; 20: 2636 - 2643.

[22] Slonim D.K., Tamayo P., Mesirov J.P., Golub T.R., and Lander E.S. Class prediction and discovery using gene expression data. *Proceedings of the Fourth Annual Conf. on Computational Molecular Biol. (RECOMB) 2000,* 263-272.

[23] Spellman, P.T. et al. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces Cerevisiae by microarray hybridization. Mol Biol Cell 9, 3273-97, 1998.

[24] Stutz J. and Cheeseman P. (1995) Bayesian Classification(AutoClass): Theory and results. Advances in Knowledge Discovery and Data Mining, 153-180, Menlo Park, CA, AAAI Press.

[25] Wu L.F., Hughes T.R., Davierwala A.P., Robinson M.D., Stoughton R. and Altschuler S.J. (2002). Large-scale Prediction of Saccharomyces Cerevisiae Gene Function Using Overlapping Transcriptional Clusters. *Nature Genetics* 31:255-265.

[26] Kerr MK, Churchill GA. Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments. Proc Natl Acad Sci USA. 2001 July; 98(16):8961-5.

[27] McShane LM, Radmacher MD, Freidlin B, Yu R, Li MC, Simon R. Methods for assessing reproducibility of clustering patterns observed in analyses of microarray data. Bioinformatics. 2002 Nov;18(11):1462-9.

Data Cleaning Using Belief Propagation

Fang Chu Yizhou Wang D.Stott Parker Carlo Zaniolo

University of California, Los Angeles
Computer Science Department
{fchu,wangyz,stott,zaniolo}@cs.ucla.edu

ABSTRACT

Effective data cleaning is critical in many applications where the quality of data is poor due to missing values or inaccurate values. Fortunately, a wide spectrum of applications exhibit strong dependencies between data samples, and such dependencies can be used very effectively for cleaning the data. For example, the readings of nearby sensors are generally correlated, and proteins interact with each other when performing crucial functions. We propose a data cleaning approach, based on modeling data dependencies with Markov networks. Belief propagation is used to efficiently compute the marginal or maximum posterior probabilities, so as to infer missing values or to correct errors. To illustrate the benefits and generality of the technique, we discuss its use in several applications and report on the data quality and improvements so obtained.

1. INTRODUCTION

Improving data quality is the key for the success of many applications including data mining. This paper focuses on two data cleaning tasks: filling in missing values and disambiguating. These two tasks can be treated in a unified framework using belief propagation.

The observation essential to the solution is that a wide spectrum of scenarios exhibit dependencies between data samples. For example, the readings of nearby sensors are generally correlated, and proteins interact with each other when performing crucial functions. Such dependencies can be exploited for cleaning the data. We illustrate this using two cases.

Incomplete Data

In sensor networks, where probing has to be minimized due to power restrictions, data is often incomplete or somewhat outdated. For popular queries such as maximum/average sensor readings, we need a cost-efficient way to infer unknown or out-of-date values while probing the sensors as little as possible.

IQIS 2005, June 17, 2005, Baltimore, MD, USA

To solve this problem, we can leverage the neighborhood relationship, since sensor readings are correlated if the sensors are geographically close. Even knowledge of far-away sensors helps, because that knowledge can be propagated via sensors deployed in between. By exploiting sensor correlation, unprobed sensors can be accurately inferred, and thus data quality can be improved.

Inaccurate Data

An example of inaccurate data is encountered in optical character recognition (OCR), a technique that translates pictures of characters into a machine readable encoding scheme. Current OCR algorithms often translate two adjacent letters " ff " into a "# " sign, or incur similar systematic errors.

In the OCR problem, the objective is not to ignore or discard noisy input, but to disambiguate legal words from recognition errors. The error patterns in OCR may be related to the shape of individual characters, the adjacency of characters, or illumination and positions. It is thus possible to correct a substantial number of errors with the knowledge of neighboring characters.

A common characteristic of the above two scenarios is the presence of data dependency. We propose a unified approach that infers missing values or disambiguating by exploiting this data dependency. We first model local data dependency by means of a graphical model, and then use *Belief Propagation* [12] for efficient inference.

In section 2, we briefly describe a special form of graphical models, *pairwise Markov network*, and how to do inference using belief propagation. Then we address the abovementioned examples in sections 3 and 4. Paper concludes with related work and discussions in Section 5.

2. BELIEF PROPAGATION ON MARKOV NETWORK

Markov networks have been successfully applied to many problems in different fields, such as artificial intelligence [7] and image analysis [10]. In this paper, we show that they can also provide very useful tools for data cleaning.

2.1 Markov Network

The Markov network is a type of graphical models suitable for modeling local dependencies. The Markov property [2] ensures that the joint probability can be factored—a critical property of Markov networks since it leads to efficient computations.

In many practical problems, we are mainly interested in are pairwise relationships among the variables. For this category of problem *pairwise Markov networks* can be used, and the joint probability of a graph configuration can be factored into:

$$P(\{x\},\{y\}) = \frac{1}{Z}\prod_{(i,j)}\psi_{ij}(x_i,x_j)\prod_i \phi_i(x_i,y_i) \quad (1)$$

where $\{x_i\}$ are random variables denoted by nodes in the Markov network, $\{y_i\}$ the evidence of the corresponding random variables, Z a normalization factor. Compatibility function $\psi_{ij}(x_i,x_j)$ captures the "internal binding" between two neighboring nodes i and j, and compatibility function $\phi_i(x_i,y_i)$ can be interpreted as "external potential" from the external field. The product over (i,j) runs over all compatible neighbors.

Solving a Markov network involves two phases:

- *The learning phase,* a phase that builds up the graph structure of the Markov network, and learns the two types of potential functions, $\phi()$'s and $\psi()$'s, from the training data.
- *The inference phase,* a phase that estimates the marginal posterior probabilities or the local maximum posterior probabilities for each random variable, such that the joint posterior probability is maximized.

  The inference phase can be solved using a number of methods, such as simulated annealing [4], mean-field annealing [8]. These methods either take an unacceptably long time to converge, or make oversimplified assumptions such as total independence between variables. We choose to use the Belief Propagation (BP) method, which has a computation complexity proportional to the number of nodes in the network, assumes only local dependencies, and has proved to be effective on a broad range of Markov networks.

2.2 Inference by Belief Propagation

Belief propagation (BP) is a powerful inference tool on Markov networks. For Markov chains and Markov networks without loops, BP is an exact inference method. Even for loopy networks, BP has been successfully used in a wide range of applications [5]. We give a short description of BP in this subsection.

The BP algorithm iteratively propagates "messages" in the network. Messages are passed between neighboring nodes only, ensuring the local constraints, as shown in Figure 1. The message from node i to node j is denoted as $m_{ij}(x_j)$, which intuitively tells how likely node i thinks that node j is in state x_j. The message $m_{ij}(x_j)$ is a vector of the same dimensionality as x_j.

There are two types of message passing rules:

- *SUM-product rule*, that computes the marginal posterior probability.
- *MAX-product rule*, that computes the maximum a posteriori probability.

For discrete variables, messages are updated using the

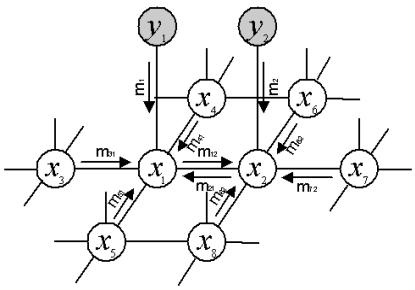


Figure 1: Message passing in a Markov network. Messages are defined by Eqs.(2) or (3) under two types of rules, respectively.

SUM-product rule:

$$m_{ij}^{t+1}(x_j) = \sum_{x_i}\phi_i(x_i,y_i)\psi_{ij}(x_i,x_j)\prod_{k\in N(i),k\neq j} m_{ki}^t(x_i) \quad (2)$$

or the MAX-product rule,

$$m_{ij}^{t+1}(x_j) = \max_{x_i}\phi_i(x_i,y_i)\psi_{ij}(x_i,x_j)\prod_{k\in N(i),k\neq j} m_{ki}^t(x_i) \quad (3)$$

where $m_{ki}^t(x_i)$ is the message computed in the last iteration of BP, k runs over all neighbor nodes of i except node j.

BP is an iterative algorithm. When messages converge, the final belief $b(x_i)$ is computed. With the SUM-product rule, $b(x_i)$ approximates the marginal probability $p(x_i)$, defined to be proportional to the product of the local compatibility at node i ($\phi(x_i)$), and messages coming from all neighbors of node i:

$$b_i(x_i)_{SUM} = x_i\phi_i(x_i,y_i)\prod_{j\in N(i)} m_{ji}(x_i) \quad (4)$$

where $N(i)$ is the neighboring nodes of i.

If using the MAX-product rule, $b(x_i)$ approximates the maximum a posterior probability:

$$b_i(x_i)_{MAX} = \arg\max_{x_i}\phi_i(x_i,y_i)\prod_{j\in N(i)} m_{ji}(x_i) \quad (5)$$

For more theoretical details of the belief propagation and its generalization, we refer the reader to [12].

3. COST-EFFICIENT SENSOR PROBING

A key research issue in sensor networks is how to minimize communication. The challenge is how to probe a small number of sensors, and to effectively infer the unprobed sensors from the known ones.

Our approach here is to model a sensor network with a pairwise Markov network, and use BP to do inference. Each sensor is represented by a random variable in the Markov network. Sensor neighborhood relationships are determined by spatial positions. For example, one can specify a distance threshold so that sensors within the range are neighbors. Neighbors are connected by edges in the network.

In the rest of this section, we study a rainfall sensornet distributed over Washington and Oregon [6]. The sensor recordings were collected during 1949-1994. We use 167 sensor stations which have collected complete recordings during that period.

3.1 Problem Description

Since rain is a seasonal phenomena, we split the data by week and build a Markov network for each week.

We need to design the potential functions $\phi_i(x_i, y_i)$ and $\psi_{ij}(x_i, x_j)$ in Eq. (1) in order to use belief propagation. One can use Gaussian or its variants to compute the potential functions. But, in the sensornet we study, we find that the sensor readings are overwhelmed by zeroes, while non-zero values span a wide range. Clearly Gaussian is not a good choice for modeling such skewed data. Neither are mixtures of gaussian, due to limited data. Instead, we prefer to use discrete sensor readings in the computation. The way we discretize data is given in section 3.3.

The $\phi()$ functions should tell how likely we observe a reading y_i for a given sensor x_i. It is natural to use the likelihood function:

$$\phi_i(x_i, y_i) = P(y_i|x_i) \tag{6}$$

The $\psi()$ functions specify the dependence of sensor x_j's reading on its neighbor x_i.

$$\psi_{ij}(x_i, x_j) = P(x_j|x_i) \tag{7}$$

3.2 Problem Formulation

We give a theoretical analysis of the problem here. As we will see shortly, the problem fits well into the maximum a posteriori (MAP) estimation on a Markov chain solvable by belief propagation.

Objective: MAP

Let X to be the collection of all underlying sensor readings, Y the collection of all probed sensors. Using Bayes' rule, the joint posterior probability of X given Y is:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{8}$$

Since $P(Y)$ is a constant over all possible X, we can simplify this problem of maximizing the posterior probability to be maximizing the joint probability

$$P(X, Y) = P(Y|X)P(X) \tag{9}$$

Eq.(9) is the objective function to be maximized, which is proportional to the maximum a posteriori probability.

Likelihood

In a Markov network, the likelihood of the readings Y only depends on the variables they directly connect to:

$$P(Y|X) = \prod_{i=1}^{m} P(y_i|x_i) \tag{10}$$

where m is the number of probed sensors.

Prior

Priors shall be defined to capture the constraints between neighboring sensor readings. By exploiting the Markov property of the sensors, we define the prior to involve only the first order neighborhood. Thus, the prior of a sensor is proportional to the product of the compatibility between all neighboring sensors:

$$P(X) \quad \propto \quad \prod_{(i,j)} P(x_j|x_i) \tag{11}$$

Solvable by BP

By replacing Eqs.(10) and (11) into the objective Eq.(9), we have the joint probability to be maximized:

$$P(X, Y) = \frac{1}{Z} \prod_{(i,j)} P(x_j|x_i) \prod_{i=1}^{N} P(y_i|x_i) \tag{12}$$

Looking back at the $\phi()$ and $\psi()$ functions defined in Eqs.(6) and (7), we see that the objective function is of the form:

$$P(X, Y) = \frac{1}{Z} \prod_{(i,j)} \psi(x_i, x_j) \prod_{i=1}^{N} \phi(x_i, y_i) \tag{13}$$

where Z is a normalizing constant.

This is exactly the form in Eq.(1), where the joint probability over the pairwise Markov network is factorized into products of localized potential functions. Therefore, it is clear that the problem can be solved by belief propagation.

3.3 Learning and Inference

The learning part is to find the $\phi()$ and $\psi()$ functions for each sensor, as defined in Eqs.(6) and (7). The learning is straight-forward. We discretize the sensor readings in the past 46 years, use the first 30 years for training and the rest 16 years for testing. In the discrete space, we simply count the frequency of each value a sensor could possibly take, which is the $\phi()$, and the conditional frequencies of sensor values given its neighbors, which is the $\psi()$.

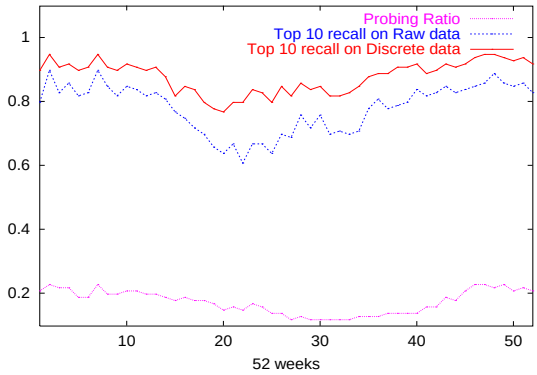
We use a simple discretization with a fixed number of bins, 11 bins in our case, for each sensor. The first bin is dedicated to zeroes, which consistently counts for over 50% of the populations. The 11 bins are assigned in a way that give roughly balanced number of readings in each bin. This very simple discretization method has been shown to work well in the sensor experiments. More elaborated techniques can be used which may further boost the performance, such as histogram equalization that gives balanced bin population with adaptive bin numbers.

For inference, belief propagation does not guarantee to give the exact maximum a posteriori distribution, as there are loops in the Markov network. However, loopy belief propagation still gives satisfactory results, as we will see shortly.

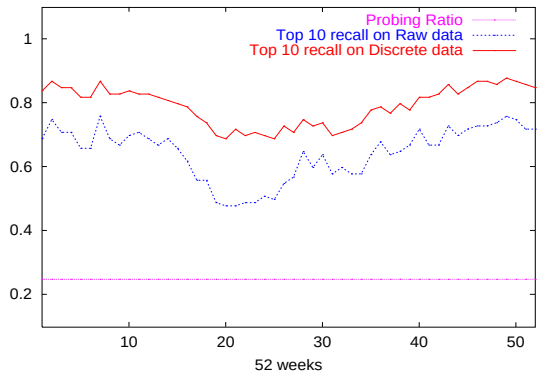
3.4 Experimental Results

Evaluation experiments are conducted on two popular types of aggregate queries: *Top-K* and *Average*. A *Top-K* query asks for the K sensors with the highest values, and *Average* asks for the averaging value of all sensor readings. These are not only popular aggregation queries that the sensor community is interested in, but also provide a good metric for probing strategies since the exact answer requires contacting all sensors.

We design probing strategies for *Top-K* and *Average* in which sensors are picked for probing based on their local maximum a posteriori probability computed by belief propagation. For comparison purposes, we also experiment with naive probing strategies that rely solely on historic statistics.



(a) BP-based *Top-K* queries.



(b) Naive *Top-K* queries.

Figure 2: *Top-K* recall rates vs. probing ratios. On average, BP-based approach probed 8% less, achieves 13.6% higher recall rate for raw values, and 7.7% higher recall rate for discrete values.

- ***BP-based Top-K probing:*** Initially, a small number of sensors with the highest expected readings are probed, based on historic statistics. After probing, all sensor reading expectations are updated using belief propagation. Then, top sensors are selected among the unprobed sensors, based on updated expectations. This updating-probing procedure iterates until beliefs of the network converge, or more than 25% of the sensors have been probed. The top K with the highest expectation (or known readings) are returned.

- ***Naive Top-K probing:*** The top 25% sensors are probed, based on historic statistics. The top K among these probed sensors are returned.

- ***BP-based Average probing:*** Initially, a small number of sensors with the largest variation are probed, based on historic statistics. After probing, all sensor reading expectations are updated using belief propagation. Then, sensors having the largest variations are selected among the unprobed sensors, based on updated expectations. This updating-probing procedure iterates until more than 25% of the sensors have been probed. Average is computed based on the final expectations and the probed readings.

- ***Naive Top-K probing:*** The 25% sensors with the largest variations are probed, based on historic statistics. The average is computed based on historic expectations and the probed readings.

Top-K query results are compared in Figure 2 (a) and (b). Experimental results are obtained for each testing day, and daily results are grouped and averaged for all days that fall into the same week of year. The results shown are for $K = 10$, but the relative performance remains the same for other values of K. In each diagram, the bottom curve shows the probing ratio, and the top two curves show the recall rates for raw values and discrete values. Discrete recall demonstrates the effectiveness of BP, while raw recall may be of more interest for real application needs. We use the standard formula to compute recall rate. Let S denotes the top-K sensor set returned, and T the true top-K set, then: $Recall = \frac{|S \quad T|}{|T|}$.

Figure 2 shows clearly that the BP-based approach outperforms the naive approach in terms of both recall rates, while requiring less probing. On average, the BP-based approach has a discrete recall of 88% and a raw recall of 78.2%, after probing only 17.5% sensors. The naive recall has a discrete recall of only 79.3%, a raw recall of only 64.6%, after probing 25% sensors.

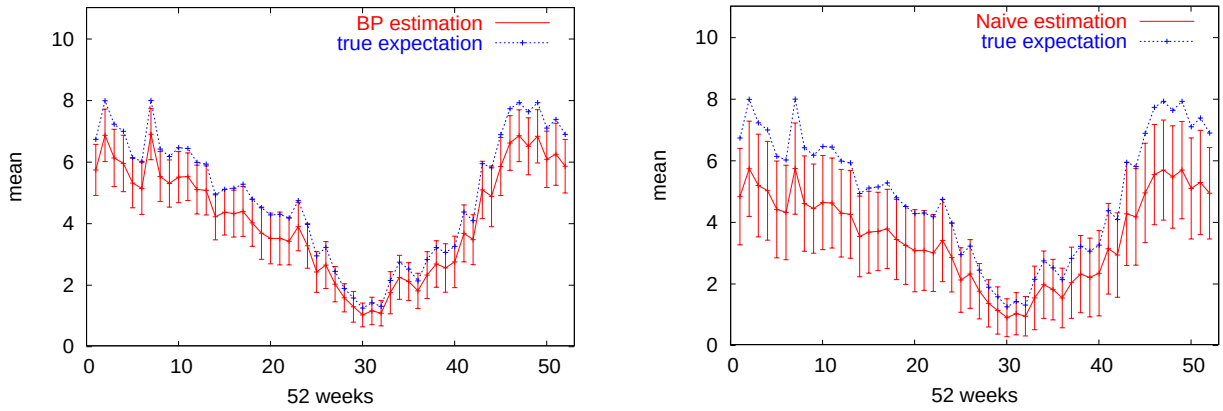
Average query results are compared in Figure 3 (a) and (b). In each diagram, the true means and the estimated means are shown, with error bars indicating one standard deviation away from the estimation in both directions. The results are obtained on discrete sensor readings which range from 0 to 11. On average, BP-based estimation has an error of -0.75 and deviation of 0.79, while Naive estimation has an error of -1.39 and deviation of 1.35. BP-based strategy is clearly much more effective at pinpointing the most uncertain sensors for probing, thus yielding more accurate results.

4. SEQUENCE DATA DISAMBIGUATING

Sequences are ordered lists of elements, such as text strings, DNA sequences, or binary codes in channel transmission. This type of data often exhibits dependencies between adjacent elements. For example, there are rich dependencies embedded in English text. This sequence data can be modeled using Markov chains—a degenerate form of Markov networks.

Moreover, errors in sequence data often have neighborhood patterns. OCR discussed in Section 1 gives an example where errors are related to the shapes of characters and to their relative positions. The mutation of a nucleotide is also influenced by its nearby bases. The fact that the Markov property is satisfied by both the sequence data itself and by errors strongly suggests that belief propagation is effective for sequence data denoising. Actually for Markov chains, belief propagation is theoretically guaranteed to give exact marginal or maximum a posteriori probabilities.

In the rest of the section, we study a problem of correcting errors in noisy documents. Despite a simple problem, it exemplifies many basic characteristics in sequence data denoising.



(a) BP-based *Average* queries. (b) Naive *Average* queries.

Figure 3: *Average* estimation error bars, on discrete sensor readings $(0-11)$. On average, BP-based estimation has an error of -0.75 and deviation of 0.79, while Naive estimation has an error of -1.39 and deviation of 1.35.

4.1 Problem Description

A document is a text sequence consisting of characters from an alphabet, while a noisy document is the result from some recognizer with systemic errors. We split the sequence into small segments, each having n characters, and let neighboring segments overlap by m characters, $m < n$. We use a random variable $x_i = (x_i^{(1)}, \cdots, x_i^{(n)})$ to represent each underlying clean segment i. The corresponding observed segment in the noisy document is denoted as $y_i = (y_i^{(1)}, \cdots, y_i^{(n)})$. Each segment, except those that start or end the sequence, has a neighbor segment on both sides.

Now we design the potential functions $\phi_i(x_i, y_i)$ and $\psi_{ij}(x_i, x_j)$. For $\phi()$, the definition should specify how likely we observe y_i given x_i. A natural choice is to define $\phi()$ to be a likelihood function

$$\phi_i(x_i, y_i) = P(y_i|x_i) \tag{14}$$

For a short segment, we can assume independence between characters. Thus, $\phi()$ can be written as

$$\phi_i(x_i, y_i) = \prod_{l=1}^{n} P(y_i^{(l)}|x_i^{(l)}) \tag{15}$$

For $\psi()$, the definition should specify how compatible two neighboring segments x_i and x_j are. Again, we can assume independence between the characters in the two segments, except for those in the overlapping part. Consider two overlapping characters, $x_i^{(k)}$ and $x_j^{(l)}$. If the probability is zero that $x_i^{(k)}$ will change to $x_j^{(l)}$ or vice versa, then the two segments, x_i and x_j, are incompatible. The resulting mutation probability of the overlapping segment quantifies the compatibility of two neighboring segments. Non-adjacent segments are incompatible. Formally, we define an asymmetric $\psi()$ function on x_i and x_j, when x_i is the left neighbor of x_j:

$$\psi_{ij}(x_i, x_j) = \prod_{l=1}^{m} P(x_j^{(l)}|x_i^{(n-m+l)}) \tag{16}$$

4.2 Learning and Inference

The objective of the learning phase is to find the $\phi()$ and $\psi()$ functions. For this purpose, we build a mutation matrix M. Each matrix element $m(i, j)$ is the unconditional mutation probability from the i-th character to the j-th: $m(i, j) = P(ch_j|ch_i)$. This can be easily computed from the training set, which consists of pairs of clean and noisy documents.

We partition the clean document and the noisy documents in the same way. The $\phi()$ of each pair of clean and observed segments is given in Eq.(14), and the $\psi()$ of each pair of neighboring clean segments is given in Eq.(16).

In inference, a subproblem is to find candidate underlying segments for a given observed segment. One can enumerate all possible candidates using the mutation matrix. But this method not only can generate too many candidates, but also ignores the following valuable information from the training data: only certain combinations of segments are possible. We restrict the candidates to be the top matches among all training segments. When the number of matches is too small, we generate some extras using the transition matrix. By doing so, we actually explore the intra-segment constraints, which are fine details that the Markov chains cannot model, as they are on the scale of segments.

4.3 Experimental Results

We choose two CVPR conference papers on the same topic: motion modeling. Both documents are distorted, using the probabilistic mutation rules in Table 1, to form pairs that consist of a clean document and a noisy document. One pair is used to train the potential functions, while the other is used for testing. For simplicity, we change all capitals into lower-case letters, replace all punctuation marks other than commas and periods into commas, and remove all figures, tables and equations. The transformed documents belongs to an alphabet of size 38 (consisting of 26 letters, 10 digits, a comma and a period).

A variety of distortion rules are used: unconditional mutation rules and k-order conditional mutation rules, $k = 1, 2, 3$. (A k-order conditional mutation depends on k neighbors on either side.) To compute the potential functions, all we need to learn is a 38 by 38 mutation matrix M for unconditional

rule	mutation prob.	# errors	% corrected
x → k	100%	56	91%
f → f	42%	-	-
f → d	30%	123	92%
f → z	28%	118	87%
th → th	48%	-	-
th → tn	52%	220	96%
se → se	36%	-	-
se → ue	18%	51	93%
se → le	25%	69	94%
se → ie	21%	58	95%
tio → tio	29%	-	-
tio → tho	20%	35	100%
tio → txo	20%	35	100%
tio → two	31%	57	98%
total words/errors: 3459/822		overall accuracy: 94%	

Table 1: Distortion rules and error correction results. Columns 1 and 2 give the rule and mutation rate, column 3 the actual number of times a rule applies, and column 4 the percentage corrected by BP inference.

mutation rates only. Yet, we are able to catch and to correct most of the mutation errors, including the higher order conditional errors. The overall correction rate is 94%, and the correction rates for high-order conditional errors are even higher, as shown in Table 1.

To help give an intuitive idea about how dependencies between text segments can be used effectively for error correction, we enclose a paragraph of distorted text here, followed by the corrected version. The misspelled words are underlined.

Distorted text:

introductxon. natural scenes contain rich stochastic mothon patterns which are characterized by the movement od a large number od distinguishable or indistinguishable elements, such as falling snow, zlock of birds, river waves, etc. tnele mothon patterns, called tektured, motion temporal tekture and dynamic tektures in the literature, cannot be analyzed by conventwonal optical zlow dields and have stimulated growing interests in both graphics and vision.

Text after disambiguating:

introduction. natural scenes contain rich stochastic motion patterns which are characterized by the movement of a large number of distinguishable or indistinguishable elements, such as falling snow, zlock of birds, river waves, etc. tnese motion patterns, called textured motion, temporal texture and dynamic tektures in the literature, cannot be analyzed by conventional optical flow fields, and have stimulated growing interests in both graphics and vision.

5. RELATED WORK AND DISCUSSIONS

Techniques for improving data quality proposed in the literature have addressed a wide range of problems caused by noise and missing data. For better information retrieval from text, data is usually filtered to remove noise defined by grammatical errors [9]. In data warehouses, there has been work on noisy class label and noisy attribute detection based on classification rules [13] [11], as well as learning from both labeled and unlabeled data by assigning pseudo-classes for the unlabeled data [1] using boosting ensembles. Most of this previous work has its own niche concerning data quality. Our work is more general in that it exploits local data constraints using Markov networks and belief propagation.

A pioneering work in sensor networks, the BBQ system [3] has studied the problem of cost-efficient probing. However, their method relies on a global multivariate Gaussian distribution. Global constraints are very strict assumptions, and are not appropriate in many practical scenarios.

The primary contribution of this paper is to propose a unified approach to filling in missing values and disambiguating, by exploiting extensive data dependencies that are present in practical problems. The techniques here proposed can be very useful for data-intensive systems, as demonstrated by our studies of real-life applications.

6. REFERENCES

[1] K. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. In *Proc. of the 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, pp. 289-296*, 2002.

[2] R. Chellappa and A. Jain. *Markov Random Fields: Theory and Application*. Academic Pr., 1993.

[3] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proc. of the 30th Int'l Conf. on Very Large Data Bases (VLDB 04)*, 2004.

[4] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. In *Science, vol. 220, no.4598*, 1983.

[5] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: an empiricial study. In *Proc. Uncertainty in AI*, 1999.

[6] University of Washington. http://www.jisao.washington.edu/data_sets/widmann/.

[7] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann publishers, 1988.

[8] C. Peterson and J. Anderson. A mean-field theory learning algorithm for neural networks. In *Complex Systems, vol.1*, 1987.

[9] G. Salton and M. McGill. *Introduction to modern information retrieval*. McGraw Hill, 1983.

[10] R. Schultz and R. Stevenson. A bayesian approach to image expansion for improved definition. In *IEEE Trans. Image Processing, 3(3), pp. 233-242*, 1994.

[11] Y. Yang, X. Wu, and X. Zhu. Dealing with predictive-but-unpredictable attributes in noisy data sources. In *Proc. of the 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD 04)*, 2004.

[12] J. Yedidia, W. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems (NIPS), Vol 13, pp. 689-695*, 2000.

[13] X. Zhu, X. Wu, and Q. Chen. Eliminating class noise in large datasets. In *Proc. of the 20th Int'l Conf. Machine Learning (ICML 03)*, 2003.

Data Quality Inference

Raymond K. Pon and Alfonso F. Cárdenas
UCLA Computer Science
Boelter Hall 4829
Los Angeles, CA 90095
(310) 825-1770
{rpon, cardenas}@cs.ucla.edu

ABSTRACT

In the field of sensor networks, data integration and collaboration, and intelligence gathering efforts, information on the quality of data sources are important but are often not available. We describe a technique to rank data sources by observing and comparing their behavior (i.e., the data produced by data sources) to rank. Intuitively, our measure characterizes data sources that agree with accurate or high-quality data sources as likely accurate. Furthermore, our measure includes a temporal component that takes into account a data source's past accuracy in evaluating its current accuracy. Initial experimental results based on simulation data to support our hypothesis demonstrate high precision and recall on identifying the most accurate data sources.

1. INTRODUCTION

A major aspect of data provenance is the ability to track the quality of data as it is processed by various transformations, each with an associated computational or intrinsic data collection error. It is important to choose trustworthy data sources when querying over multiple data sources [1]. Users of data warehouses regard the quality of information as important and as a factor in measuring the utility of a data warehouse [2]. Furthermore, the inclusion of data quality information has an impact on decision-making and decision-support systems as well [3, 4]. Also data conflicts, occurring when heterogeneous data sources are integrated, can be resolved by considering the quality of the data sources involved [5]. The accuracy of data can also be used to rank query results as well (as opposed to the relevance of query results to the query) [6]. Clearly, this trustworthiness and quality information should be stored as part of a data item's provenance. The following is a general query in which data quality can be used to answer:

Query 1: *"Given many genomic databases where data has been collected by various means and institutions, find a DNA sequence that satisfies a condition C."* In this query, collections of data sets (i.e., DNA sequences) have been collected by different instruments and/or possibly derived by various and possibly multiple transformations. Each of these instruments and transformations has a different degree of reliability and error. Additionally, the data sets that are relevant to the query may be numerous, so it is necessary to rank data sets by their "quality" or trustworthiness (i.e., how reliable the data sets are).

IQIS 2005, June 17, 2005, Baltimore, MD, USA.

However, it is unclear as to where metadata regarding data quality comes from. User-provided ratings of data sets or sources can be used to rate the quality of data sources [1], but is clearly a subjective measure and would require large samples to get any meaningful results. Error measures can be provided by data sources providers along with data sets, but may be inaccurate, difficult to use, incomplete, or untrustworthy [7]. Thus, we describe a technique to rank data sources by observing and comparing the behavior (i.e., the data produced by data sources) to rank them in terms of their quality. Intuitively, in our measure, data sources that agree with accurate data sources are likely to be accurate. Furthermore, in our measure, data sources that have been accurate in the past are also likely to be accurate in the future. We provide some initial experimental results based on simulation data to support our hypothesis. The following sub-sections discuss motivations for this work and the related works. In section 2, we describe our technique for data source ranking. In section 3 and 4, we present our initial experimental results and possible roads of future research, respectively.

1.1 Motivation

We describe three possible application areas in which the modeling of data accuracy and trustworthiness are important.

Application 1: Sensor networks are becoming increasingly prevalent in observing wildlife, monitoring environmental conditions, monitoring of soldiers in the field, and the detection of harmful biological and chemical agents, with practical applications in homeland security [8]. The effectiveness of these sensor networks is highly dependent on the accuracy of the networks, which is a function of the current battery level of the device, interference, and intrinsic error in data collection. For example, in the near future, soldiers may be equipped with data capturing devices, making each soldier a sensor [9], to give field commanders current battlefield status reports. Data captured by soldiers may be conflicting and/or erroneous because of the human element involved in the data capturing process. It would be advantageous to be able to determine the more trustworthy "sensors" in capturing the current situation to filter out noisy data and to reduce the consumption of resources (e.g., manpower, time, and battery-life).

Application 2: In biomedical research, research facilities frequently collaborate with each other, sharing experimental data and results. In particular, comparing genome sequences from different species has become an important tool for identifying functions of genes [10]. This necessitates dynamically integrating different databases or warehousing them into a single repository. Scientists need to know how reliable the data is if they are to base their research on it. Pursuing incorrect theories cost time and

money. The obvious solution to ensure data quality is curation, but data sources are autonomous and as a result sources may provide excellent reliability in one area, but not in all data provided, and curation slows the incorporation of data. Data providers will not directly support data quality evaluations to the same degree since there is no equal motivation for them to and there are no standards in place for evaluating and comparing data quality [7]. Thus, automatic, impartial, and independent data quality evaluation would be needed.

Application 3: In intelligence gathering efforts, data is often collected from many heterogeneous data sources, such as satellites, human assets, transcripts, wiretaps, etc. It is obvious that each of these data sources have different degrees of quality and trust. And with the multitude of data sources to incorporate, it is currently time-consuming to sift through each of these data sources to determine which the most accurate sources are. To make the correct decisions based on the intelligence available in a timely manner, we will need an automatic means to determine accurate data sources and to be able to detect malicious or compromised data sources to prevent them from influencing decision-making processes.

1.2 Related Works

There has been a significant amount of work in the area of information quality, ranging from techniques in assessing information quality and accuracy to building large-scale data integration systems over heterogeneous data sources. For example, the DaQuinCIS system [11-13] is a cooperative information system where data source providers are evaluated by data source users in a peer-to-peer system. Unfortunately, such a system relies heavily on the participation of users in the review of the quality of data in the system, which may not be practical in real-life environments. Users may not reliably or consistent in evaluating data sources.

Other works have taken other approaches in modeling and capturing data quality. Some have developed data models to model data quality but rely on data quality metadata being available, such as data sources publishing such information [14-19]. Unfortunately, these approaches rely on precise and accurate metadata. However, such metadata are not always available [7] and there is no single agreed upon standard in describing data lineage. Additionally, it may be possible for malicious processes to corrupt or "spam" query results by providing false metadata.

There has also been work in methodologies in assessing of the quality of data in databases [20-23]. However, such methodologies rely on human assessment of the data, which is often time-consuming and possibly error-prone.

Previous works have assumed that the metadata regarding the quality of data is available, accurate, and unbiased, either published by the data providers themselves or provided by user-rankings of the data sources. Our contribution is that we do not assume that such metadata is available and reliable. Rather, our automated approach examines how well the data sets produced by data sources agree with one another, and infer the rankings of the data sources in terms of their accuracy. We take an approach similar to Google's PageRank [24]. Instead of evaluating the popularity of web sites by measuring how many other popular websites link to them as in PageRank's approach, we evaluate the accuracy of data sources by measuring how well other data sources agree with the data they produce. This approach is automated, does not rely on possibly faulty and limited metadata, and does not require human assessment.

2. DATA SOURCE RANKING

Traditionally, the ranking of query results was based on the relevance of a user's query. However, the quality of the results could be improved if we incorporated a data quality measure in addition to their relevance to the user's query.

We wish to do following in general:

1. Rank the data sets or data sources in order of their accuracies.
2. Determine the top-k accurate data sets or data sources.

This ordering is important particularly in data integration systems, where there are numerous data sources available of varying accuracy that changes dynamically across time. Ideally, given a query, we would like to contact each data source; however, this may be prohibitively expensive if there are budget constraints such as time and network resources. Such applications can be found in sensor networks, where battery-life is limited, and intelligence-gathering efforts, where manpower and time are limited. Thus, it would be advantageous to determine the most accurate set of data sources, so that they can be contacted in answering a query. Additionally, this methodology could be used for identifying malicious or compromised data sources that are attempting to feed false information into the data integration system. We provide the following framework for ranking the accuracy or trustworthiness of data sources based on observing and comparing data source behavior without any a priori knowledge of their relative accuracies to help solve the problem. In our model, we assume that schema and data heterogeneity have been reconciled, which is beyond the scope of this work.

2.1 General Framework

Let D be a set of data sources. A data source $d_i \in D$ generates a table $T_i^t(k,v)$ for a query Q where t is the time index, k is the key column, and v is the value column of the table. We want to derive a metric $A_i^t \in [0,1]$ that measures the relative accuracy of data source d_i at time t such that $A_i^t < A_j^t$ if d_i is less accurate than d_j at time t. We define such a metric as the weighted average of the previous accuracy estimate at time index $t-1$ and the accuracy estimate derived by observing the data generated by data sources in D:

$$A_i^t = h(t)A_i^{t-1} + (1-h(t))c(i,t) \quad (1)$$

The intuition behind A_i^t is that a data source's accuracy should be a function of its past accuracy (i.e., reputation) and its current behavior. The function $h(t)$, where $0 \le h(t) \le 1$, is the historical weight function that determines the contribution of the accuracy estimate at the previous time index. The intuition behind the historical component of the accuracy measure is that a data source that has been accurate (or inaccurate) in the past should also be accurate (or inaccurate) in the near future. For simplicity, the above historical component assumes a Markovian behavior in the evolution of the data sources, where the accuracy at time t is only dependent on the value at t-1. However, it will be interesting to see if we can improve the quality of our estimation by taking into

account a sliding window of size w: $[A_i^t, A_i^{t-1}, \ldots, A_i^{t-w}]$. Thus, the historical component would consist of a weighted sum of all accuracy estimates within the last w time indexes, where each estimate is weighted with a decaying weight function. The decaying weight function would assign a higher weight to more recent estimates than older estimates. A sliding window version of the equation (1) would be of the following form, where *w(j, t – 1)* is the decaying weight function:

$$A_i^t = h(t) \sum_{j=t-w}^{t-1} w(j,t-1) A_i^j + (1-h(t)) c(i,t) \qquad (2)$$

However, we leave this issue to future research, where we will study the most appropriate decaying weight function and the optimal sliding window size.

The cohesion function *c(i,t)* determines the new accuracy estimate by observing data generated by data sources in D at the current time index and how well each data source agrees with one another. The cohesion function *c(i,t)* that we propose is the following:

$$c(i,t) = f(i,t) + \frac{(1-f(i,t))}{|D|-1} \sum_{d_j \in D-\{d_i\}} a(i,j,t) c(j,t) \qquad (3)$$

The function *a(i,j,t)* is the agreement function, which outputs 0 when data sources d_i and d_j are in strong disagreement regarding the data in T_i^t and T_j^t, and outputs 1 when d_i and d_j strongly agree, and values between 0 and 1 for other levels of agreement. The intuition behind *c(i,t)* is:

- If a data source agrees with an accurate data source, it should also be accurate.
- If a data source agrees with an inaccurate data source, it should also be inaccurate.
- A data source has a probability *f(i,t)* of being absolutely accurate independent of any agreement/disagreement with the other data sources.

Thus, given a system of equations of $|D|$ equations and $|D|$ variables, it is possible to determine *c(i,t)* for all $d_i \in D$. The function *f(i,t)* is the dampening factor function (similar to that defined in Google's PageRank algorithm [24]). In addition to being the probability that a data source d_i is absolutely accurate independent of its agreement with the other data sources, the function *f(i,t)* will prevent the solution to the system of equations from consisting of entirely zeroes for all *c(i,t)*.

2.2 Agreement Functions

There are several possible definitions for *a(i, j, t)*, such as the *tupleOverlap* function, which measures the proportion of tuples in approximate agreement (within some allowable difference ε) in the set of tuples whose key values are generated by both data sources d_i and d_j:

$$tupleOverlap(i,j,t) = \frac{|T_i^t \underset{\substack{T_i^t.k=T_j^t.k \\ T_i^t.v=T_j^t.v}}{\triangleright\!\triangleleft} T_j^t|}{|T_i^t \underset{T_i^t.k=T_j^t.k}{\triangleright\!\triangleleft} T_j^t|} \qquad (4)$$

Another possible definition for *a(i,j,t)* is the *cosineOverlap* function, which measures the complement of the cosine distance of two sets of data over the same key values generated by d_i and d_j:

$$cosineOverlap(i,j,t) = \frac{V(i,j,t)^T V(j,i,t)}{|V(i,j,t)||V(j,i,t)|} \qquad (5)$$

The vector *V(i,j,t)* can be roughly defined as $V(i,j,t) = \pi_{T_i^t.v}(T_i^t \underset{T_i^t.k=T_j^t.k}{\triangleright\!\triangleleft} T_j^t)$, except it is an ordered vector in which the values stored in the vector are ordered by their corresponding key values in T_i^t. There is also a Euclidian-based function for *a(i, j, t)*, which we will discuss in further detail later.

Given this system of $|D|$ equations and $|D|$ variables, we can arrange the equations to the following form:

$$A(t) * C(t) = F(t) \qquad (6)$$

A(t) is defined as the following matrix:

$$A(t) = \frac{f(t)-1}{|D|-1} \begin{bmatrix} \frac{|D|-1}{f(t)-1} & a(1,2,t) & \cdots & a(1,|D|,t) \\ a(2,1,t) & \frac{|D|-1}{f(t)-1} & \cdots & a(2,|D|,t) \\ \vdots & \vdots & \ddots & \vdots \\ a(|D|,1,t) & a(|D|,2,t) & \cdots & \frac{|D|-1}{f(t)-1} \end{bmatrix}$$

C(t) and *F(t)* are also defined as the following matrices:

$$C(t) = \begin{bmatrix} c(1,t) \\ c(2,t) \\ \vdots \\ c(|D|,t) \end{bmatrix}, \; F(t) = f(t) \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

The solution to equation (6), *C(t)*, is a vector where each entry $C(t)_i$ estimates the accuracy of data source d_i. The matrix *A(t)* can also be normalized with respect to maximum or sum of the entries in each of the rows (horizontal normalization) or in each of the columns (vertical normalization). We can horizontally normalize the matrix *A(t)* by performing the following division on every entry $A(t)_{i,j}$ in row *i*, column *j,* except for entries where *i = j*:

$$A'(t)_{i,j} = A(t)_{i,j} \Big/ Hor(t,i)$$

Hor(t,i) can either be the sum or the maximum value of all the entries in row *i* excluding the entry $A(t)_{i,i}$. We can also similarly define a function Ver(t,j) for vertical normalization ($A(t)_{i,j} = A(t)_{i,j} \Big/ Ver(t,j)$) to be either the sum or the maximum value of all entries in column *j* excluding the entry $A(t)_{j,j}$.

Given our normalization techniques, we can now discuss in further detail the Euclidian-based function for *a(i, j, t)* mentioned briefly before. Because Euclidian-distance is unbounded, normalization would be required to describe the amount of overlap or agreement. We define the Euclidian-based function *eOverlap* for *a(i, j, t)*:

$$eOverlap(i,j,t) = 1 - eDist'(V(i,j,t), V(j,i,t)) \qquad (7)$$

The function *eDist'* is simply the Euclidian distance of the vectors *V(i,j,t)* and *V(j, i, t)*, normalized in a similar manner as described above.

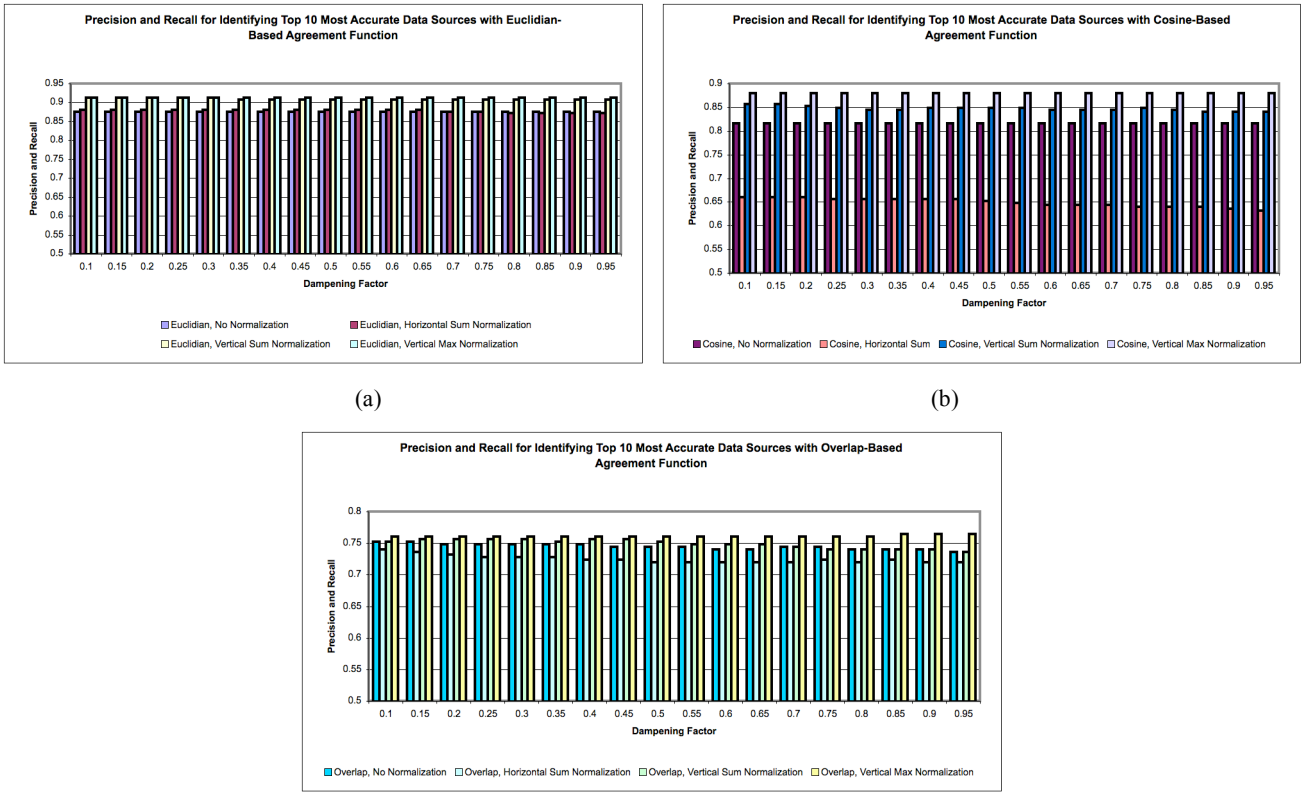


Figure 1: The precision and recall for identifying the top 10 most accurate data sources with (a) the Euclidian-based agreement functions, (b) the Cosine-based agreement functions, and (c) the Overlap-based agreement functions with ε = 0.1.

3. EXPERIMENTAL RESULTS

We hypothesize that the above framework can be used as a springboard in solving the general problem of identifying accurate data sources. To do so, we will need to identify adequate *h(t)*, *a(i,j,t)*, and *f(i,t)* functions through experimentation. Our initial experiments examine the cohesion function *c(i,t)* with a dampening factor *f* independent of time (i.e., the probability of a data source being absolutely accurate independent of all other data sources is constant), and excluding incorporation of the historical component. As a result, the combination of equations (1) and (3) reduces to the following:

$$A_i^t = c(i,t) = f + \frac{1-f}{n-1} \sum_{d_j \in D-\{d_i\}} a(i,j,t)c(j,t) \qquad (8)$$

We implemented a Java prototype, using JAMA (Java Matrix Package) [25] for solving the system of equations, and experimented on simulation data consisting of 100 data sources, each producing 20 different tuples, each consisting of a key (of type integer) and a value (of type double). In each run, a data set, consisting of 20 keys and values randomly assigned to each key with a uniform distribution, represent the "actual" data that each data source will attempt to report. Additionally, in each run, each data source was randomly assigned positive error values according to a Gaussian distribution with an average of 0 and a standard deviation of 1.0. For each run, we ran five iterations, where each data source produced a data set, consisting of values for each key, where each value is randomly generated with a Gaussian distribution with a standard deviation equal to that of the data source's error value and an average equal to that of the "actual" data item's value, essentially perturbing each data item's value with data source's error value. Data sources with large error values will generally generate values farther away from the "actual" value than data sources with smaller error values. We ran a total of five runs, consisting of five iterations, and averaged the results.

Figure 1 shows the precision and recall of the various agreement functions and normalizations as the dampening factor *f* is varied. Note that the overlap-based functions are using a difference margin ε = 0.1. The figure clearly shows that the dampening factor has very little effect in identifying the top 10 most accurate data sources. However, the figure does show that the vertical normalization with respect to the maximal value of the column yields the best performance. Additionally, the figure shows that the overlap-based functions perform the worse, with the cosine-based functions performing well and the Euclidian-based functions performing even better with a precision and recall of over 90%. The overlap-based functions suffer from having a fixed allowable difference margin that is difficult to estimate without knowing the nature of the data and the data sources a priori. The cosine-based functions perform better than the overlap-based functions because no such assumption is needed but does not accurately capture the amount of distance/overlap as Euclidian-based functions do.

To summarize, Figure 2 shows the performance of the three agreement functions with various normalizations using a fixed

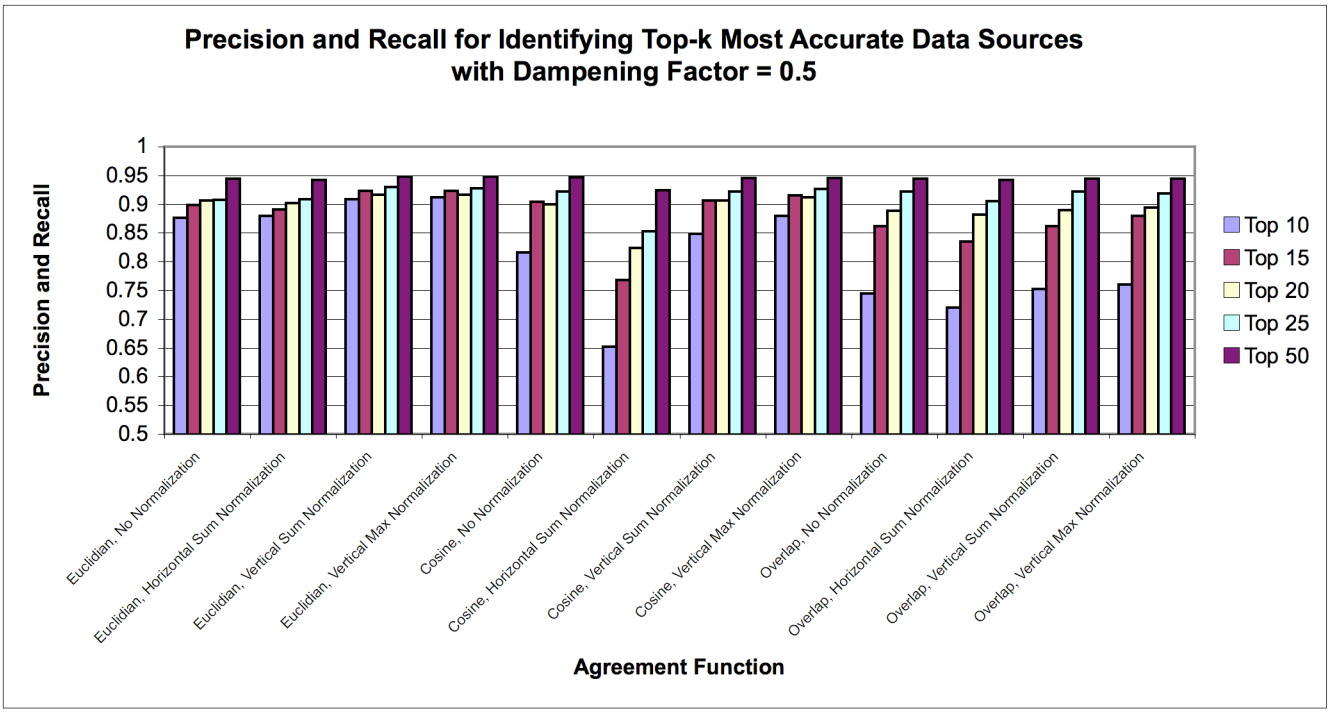


Figure 2: The precision and recall for identifying the top 10, 15, 20, 25, and 50 most accurate data sources with a dampening factor of 0.5

dampening factor of 0.5 (since current results do not definitively indicate the best value for *f*, we selected a mid-range value for *f*). The figure clearly shows that the Euclidian-based agreement function with vertical max and sum normalizations performs the best with a precision and recall of over 90%.

4. FUTURE WORK

One of the caveats of the current technique is that it relies on data sources reporting on the same set of data items. Often, it may be the case where data sources will report about different data items. Future study will have to be done to evaluate the current technique's effectiveness over incomplete and heterogeneous data sources. Additionally, the current technique may suffer from possibly expensive polling of all data sources. In future work, we will need to devise an efficient and intelligent sampling technique to alleviate such a problem while still preventing the staleness of estimates. One obvious possibility is to use the data gathered during a query (which is essentially free from the point of view of the quality estimator since such a cost will need to be incurred anyway to answer the query) to estimate a new relative accuracy measure than can be used for the next query. However, only data sources with high accuracy estimates will have their estimates updated and the estimates of data sources of low accuracy will become stale, since accurate data sources are the only data sources consistently being probed since they are selected to the answer the query. Thus, we will need to explore additional sampling techniques [26], such as polling for only small subsets of data from a majority of data sources, to solve this problem and to be able to associate a confidence metric in the ranking generated by our methodology.

Additionally, computing the solution to a set of *n* *c(i,t)* equations with *n* variables may be computationally expensive if *n* is very large. Thus, we will also explore techniques to speed up this computation with an acceptable margin of error, such as using an iterative approach, using old *c(j,t-1)* values for computing the new *c(i,t)* value in equation (3). Figure 3 shows promising preliminary results regarding the performance of the iterative solution, indicating that we can arrive to a reasonably good estimation in very few iterations and that the dampening factor has some effect on how fast we can arrive to a solution. We use an initial estimate of *c(i,-1)* = 1 for all data sources and use the Euclidian-based agreement function with vertical sum normalization while varying the dampening factor. Future work will further explore the effect of the dampening factor.

In this preliminary study, we randomly assign error values to the

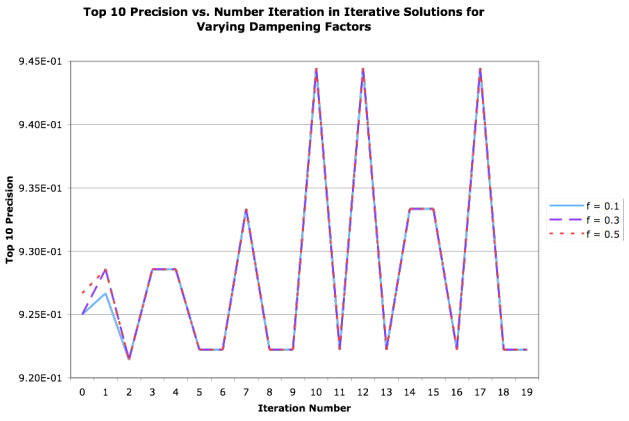


Figure 3: Performance of attaining an iterative solution using *c(i,-1)* = 1 and the Euclidian-based agreement function with vertical sum normalization.

data sources with a Gaussian distribution. Additional research will include further study on how well our cohesive function performs with other probability distributions, such as uniform distributions. We also hypothesize that such a technique can be used to automatically identify faulty or failing data sources dynamically, such as a sensor or an intelligence asset. We will need to experiment with the historical component of our accuracy measure. We will study how robust and reactive our accuracy measure will be when the accuracy of data sources becomes dynamic, as opposed to being static as in the case of this preliminary study.

Although we have experimented with an overlap-based function using a difference margin $\varepsilon = 0.1$ and could have used other values for ε to see the effect on the precision and recall of identifying the top-k most accurate data sources, the results indicate that that the overlap-based function performs poorly compared to the Euclidian and cosine-based functions with this value for ε. Another value for ε would have probably been better, but we hypothesize that the optimal ε is dependent upon the domain application of the data. In later work, we will examine the effect of ε when real-life data (e.g., sensor data) becomes readily available.

Currently, our accuracy measure evaluates the accuracy of data sources based a single domain of data (i.e., a single topic). However, data sources may provide data for multiple domains (i.e., multiple topics) and may be more accurate in one domain than another. There are two possible attitudes in approaching this problem. A "suspicious" attitude would suspect all data (regardless of topic) provided by a data source if a data source contradicts a more trustworthy data source. A "trusting" attitude would only suspect a minimal set of data (i.e., data from the contradicting topic) that contradicts a more trustworthy data source, which is a similar attitude taken in [27]. Future research will examine how these attitudes can be incorporated into the overall accuracy measure.

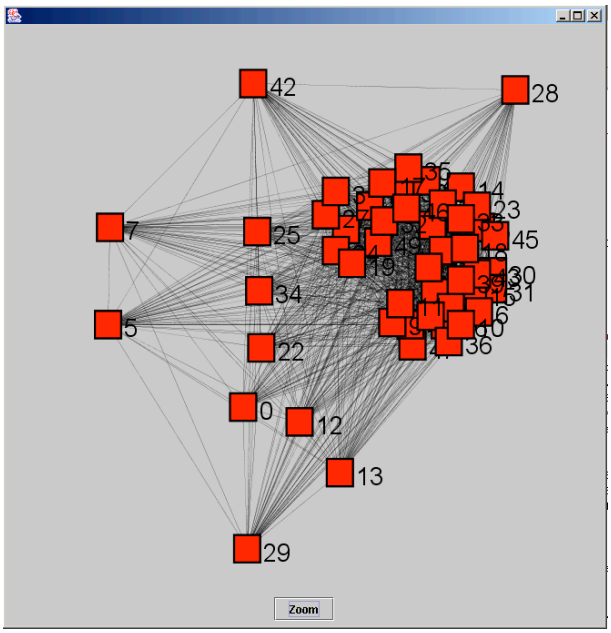


Figure 4: Network of agreeing data sources

We also envision that this technique can be used to identify communities of data sources in which members of the community share common "beliefs." In Figure 4, a graph generated with JUNG (Java Universal Network/Graph Framework) [28] consisting of 50 nodes, each representing a data source, are connected by edges, whose lengths are the Euclidian-distance of the data sets generated by the connecting nodes. It is clear from the graph that nodes that are in high agreement with one another are clustered very closely with each other; whereas, outliers in the graph disagree with the cluster and can be considered as inaccurate. Future work will include studies how clustering techniques can be used to identify communities of data sources, such as that from social network analysis [29].

5. CONCLUSION

We have presented an automated technique for inferring the quality of data sources without the luxury of metadata. Our main contribution is a framework to capture the historical accuracy of data sources and the relationship of data sources in how well they agree with one another (i.e., the cohesive function). Our second contribution is a preliminary study of the cohesive function, examining the precision and recall of identifying the top-k most accurate data sources with various agreement functions and normalizations. We have shown that the Euclidian-based agreement function vertically normalized performs the best.

We have also identified several significant challenges and future roads of research, including performance optimizations, exploring various sampling techniques, developing robust yet reactive accuracy estimations, and identifying communities of data sources.

6. ACKNOWLEDGMENT

The authors would like to thank Roderick Son, from the UCLA Medical Imaging Informatics Group, Terence Critchlow and David Buttler from the Lawrence Livermore National Laboratory, and the anonymous reviewers for their invaluable inspiration and input for this work. This work is partially funded by the National Foundation Grant # IIS 0140384.

7. REFERENCE

[1] D. Buttler, M. Coleman, T. Critchlow, R. Fileto, W. Han, C. Pu, D. Rocco, and L. Xiong, "Querying multiple bioinformatics information sources: can semantic web research help?" *SIGMOD Record*, vol. 31, pp. 59-64, 2002.

[2] A. Rudra and E. Yeo, "Issues in user perceptions of data quality and satisfaction in using a data warehouse-an Australian experience," presented at 33rd Annual Hawaii International Conference on System Sciences, 2000.

[3] I. N. Chengular-Smith, D. P. Ballou, and H. L. Pazer, "The impact of data quality information on decision making: an exploratory analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, pp. 853-864, 1999.

[4] R. A. Dillard, "Using data quality measures in decision-making algorithms," *IEEE Expert*, vol. 7, pp. 63-72, 1992.

[5] F. Naumann, "From databases to information systems - information quality makes the difference," presented at the International Conference on Information Quality (IQ 2001), Cambridge, MA, 2001.

[6] M. Gertz, M. T. Ozsu, G. Saake, and K. U. Sattler, "Report on the Dagstuhl seminar: 'data quality on the web'," *SIGMOD Record*, vol. 33, pp. 127-132, 2004.

[7] T. Critchlow, L. Liu, D. Buttler, D. Rocco, and C. Pu, "Towards Automatic Discovery and Identification of Bioinformatics Web Interfaces," [Online] Available: http://sirius.cs.ucdavis.edu/Dagstuhl03/presentations/03362.CritchlowTerence.Slides.ppt, 2003.
[8] V. Kumar (editor), "Special Issue on Sensor Network Technology and Sensor Data Management," *SIGMOD Record*, vol. 32, 2003.
[9] F. Donovan, "Army to deploy hand-held devices to make every soldier into a sensor," [Online] Available: http://www.aviationnow.com/avnow/news/channel_netdefense_story.jsp?id=news/arm04294.xml, 2004.
[10] F. S. Collins, E. D. Green, A. E. Guttmacher, and M. S. Guyer, "A vision for the future of genomics research," *Nature*, vol. 422, pp. 835-847, 2003.
[11] M. Mecella, M. Scannapieco, A. Virgillito, R. Baldoni, T. Catarci, and C. Batini, "Managing data quality in cooperative information systems," in *Lecture Notes in Computer Science 2519*, 2002, pp. 486-502.
[12] M. Scannapieco, A. Virgillito, C. Marchetti, M. Mecella, and R. Baldoni, "The DaQuinCIS architecture: a platform for exchanging and improving data quality in cooperative information systems," *Information Systems*, vol. 29, pp. 551-582, 2004.
[13] L. D. Santis, M. Scannapieco, and T. Catarci, "Trusting data quality in cooperative information systems," presented at CoopIS 2003, 2003.
[14] J. Widom, "Trio: a system for integrated management of data, accuracy, and lineage," presented at CIDR 2005, Pacific Grove, California, 2005.
[15] G. A. Mihaila, L. Raschid, and M.-E. Vidal, "Using quality of data metadata for source selection and ranking," presented at Third International Workshop on the Web and Databases, WebDB'2000, Dallax, TX, 2000.
[16] G. A. Mihaila, L. Raschid, and M.-E. Vidal, "Source selection and ranking in the websemantics architecture using quality of data metadata," *Advances in Computers*, vol. 55, pp. 87-118, 2002.
[17] M. Gertz, "Managing data quality and integrity in federated databases," presented at IFIP TC11 Working Group 11.5, Second Working Conference on Integrity and Internal Control in Information Systems: Bridging Business Requirements and Research Results, 1998.
[18] F. Naumann, J. C. Freytag, and U. Leser, "Completeness of integrated information sources," *Information Systems*, vol. 29, pp. 583-615, 2004.
[19] F. Naumann, "Quality-Driven Query Answering for Integrated Information Systems," in *Lecture Notes in Computer Science*, G. Goos, J. Hartmanis, and J. v. Leeuwen, Eds. Berlin, Germany: Springer-Verlag, 2002, pp. 166.
[20] A. Motro and I. Rakov, "Estimating the quality of databases," presented at 1996 Conference on Information Quality, Cambridge, MA, 1996.
[21] M. Bobrowski, M. Marre, and D. Yankelevich, "A homogeneous framework to measure data quality," presented at IQ 1999, Cambridge, MA, 1999.
[22] B. Pernici and M. Scannapieco, "Data quality in web information systems," presented at ER 2002, 2002.
[23] Y. W. Lee, D. M. Strong, B. K. Kahn, and R. Y. Wang, "AIMQ: a methodology for information quality assessment," *Information Systems*, vol. 29, pp. 133-146, 2004.
[24] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," presented at 7th World Wide Web Conference (WWW7), 1998.
[25] J. Hicklin, C. Moler, P. Webb, R. F. Boisvert, B. Miller, R. Pozo, and K. Remington, "JAMA: Java Matrix Package," [Online] Available: http://math.nist.gov/javanumerics/jama/, 2005.
[26] J. Cho and A. Ntoulas, "Effective Change Detection using Sampling," presented at VLDB Conference, Hong Kong, China, 2002.
[27] L. Cholvy and C. Garion, "Querying several conflicting databases," presented at ECSQARU-03 Workshop Uncertainity, Incompleteness, Imprecision, and Conflict in Multiple Data Sources, Aalborg, 2003.
[28] Jung Framework Development Team, "JUNG: Java Universal Network/Graph Framework," [Online] Available: http://jung.sourceforge.net/index.html, 2005.
[29] S. Staab, P. Domingos, P. Mika, J. Golbeck, L. Ding, T. Finin, A. Joshi, A. Nowak, and R. R. Vallacher, "Social Networks Applied," *Intelligent Systems, IEEE [see also IEEE Expert]*, vol. 20, pp. 80-93, 2005.

Author Index