Corpus-Based Knowledge Representation

Alon Y. Halevy
University of Washington
Seattle, Washington, U.S.A.
alon@cs.washington.edu

Jayant Madhavan
University of Washington
Seattle, Washington, U.S.A.
jayant@cs.washington.edu

Abstract

A corpus-based knowledge representation system consists of a large collection of disparate knowledge fragments or schemas, and a rich set of statistics computed over the corpus. We argue that by collecting such a corpus and computing the appropriate statistics, corpus-based representation offers an alternative to traditional knowledge representation for a broad class of applications. The key advantage of corpus-based representation is that we avoid the laborious process of building a (often brittle) knowledge base. We describe the basic building blocks of a corpus-based representation system and a set of applications for which such a paradigm is appropriate, including one application where the approach is already showing promising results.

1 Introduction

Declarative representation of knowledge has long been acknowledged as a key building block of AI systems. A knowledge base (KB) with associated reasoning mechanisms can serve as a backbone for tasks such as query answering, learning and diagnosis. One of the key challenges in deploying knowledge representation (KR) systems is the cost and the complexity of building the knowledge base, especially in cases where it needs to cover a broad domain. Building a KB is highly labor intensive for several reasons. First, the acquisition of domain knowledge and articulating it in a formal language can be incredibly difficult. Second, since the KB needs to form a coherent logical whole, it is hard to distribute its creation of a knowledge base over multiple experts or knowledge engineers. Finally, no matter how broad the KB, it will be brittle at the edges of its knowledge - it can only be used for tasks whose knowledge needs have been anticipated in advance. While all of the aforementioned difficulties have received significant attention, together they still remain a fundamental challenge for knowledge representation technology.

This paper argues that there is an alternative approach to knowledge representation that has the potential of providing competitive functionality for a broad class of applications. In a nutshell, this approach, which we term *corpus based representation,* is based on collecting a large corpus of disparate fragments of knowledge, and building a set of tools that are based on analyzing properties of the corpus. The fragments of knowledge in the corpus can include individual KBs, database schemas with or without data instances of the schema, queries written over KBs and databases, and any form of meta-data associated with them. Unlike a KB that needs careful ontological design, the corpus is a set independent uncoordinated contributions. The intuition is that if the corpus is large enough, then the patterns we identify in it can be of great use for knowledge intensive tasks.

Corpus-based knowledge representation is an outgrowth of our work on *schema* and *ontology matching* [Doan *et al.,* 2001; 2002]. The matching problem (which we elaborate on in Section 2), is to find a semantic mapping between two disparate representations, be they database schemas or ontologies. While a detailed KB about the domain of the representations being matched could be of great value for the matching task, such KBs rarely exist, and their construction is not cost effective. Instead, our approach to matching is based on analyzing the variations of representations in a corpus of schemas. This paper extends the idea of using a corpus to a general approach to knowledge representation. Specifically, we identify other tasks in which corpus-based representation can be useful, and then define preliminary version for the contents of a corpus and the interfaces it should support.

We emphasize that corpus-based representation is not a replacement for traditional knowledge representation. There are many tasks in which very finely tuned reasoning is required, and such reasoning can only be done with a very well designed knowledge base (e.g., medical diagnosis, monitoring spacecraft, and making sense of tax laws). Our goal is to explore the space of problems in which the laborious construction of knowledge bases can be avoided.

2 A motivating application: matching disparate representations

The discussion in this paper spans many kinds of representations, ranging from database schema to ontologies in expressive KR formalisms. Collectively, we use the term *domain model* to refer to representations in any of these formalisms. The term *schema* refers to intensional knowledge, such as the names of database tables and their respective columns, or to the terminological component of a KB. The ground facts (or, rows in a database) are called *instance data.*

Sharing data among multiple data sources and applications is a problem that arises time and again in large enterprises, B2B settings, coordination between government agencies, large-scale science projects, and on the World-Wide Web. The problem has received significant attention in both the AI and Database communities, and over the years, several paradigms for data sharing have been implemented, each appropriate for certain scenarios. Examples include data warehousing oriented architectures, data integration systems [Lenzerini, 2002; Draper *et al,* 2001], message passing systems (e.g., [MQSERIES, 2003]), web services, and peer-data management systems LHalevy *et al,* 2003b; Kalnis *et al,* 2002; Bernstein *et al,* 2002].[

No matter what paradigm we employ to share the data, a key problem is the semantic heterogeneity between the domain models of data sources that were originally designed independently. Thus, to obtain meaningful interoperation, one needs a *semantic mapping* between the schemas. A semantic mapping is a set of expressions that specify how the data in one database corresponds to the data in another database [Madhavan *et al,* 2002]. While languages for specifying semantic mappings have been developed and are well understood (see [Lenzerini, 2002; Ha^{1}evy, 2001] for surveys), the creation of semantic mappings has become a key bottleneck as it is labor intensive and error-prone.

The goal of schema matching is to assist a human to relate two domain models. Complete automation of the process is unlikely to be possible, but the goal is to significantly increase the productivity of human experts. The matching problem is difficult because it requires understanding the underlying semantics of the domain models being matched. While a domain model (with its instance data) provides many clues on its intended semantics, it does not suffice in order to relate it to a different domain model.

The process of generating a semantic mapping has traditionally been divided into two phases. The first phase finds a *match* between the two schemas. The match result is a set of *correspondences* between elements in the two schemas, stating that these elements are *somehow* related. For example, a correspondence may state that agent in one domain model corresponds to contact in another. The second phase builds on the correspondences by creating the mapping expressions. The mapping expressions, often expressed as queries or rules, enable translating data from one data source to another, or reformulating a query over one data source into a query on the other. A plethora of techniques have been proposed for schema matching: see [Rahm and Bernstein, 2001] for a survey, and [Noy and Musen, 2002; Doan *et al,* 2002; Do and Rahm, 2002] for some work since then. Collectively, these techniques mirror the heuristics that a human designer may follow. For example, techniques have considered exploiting relationships between names of elements in the domain models, structural similarities between them, similarities in data values, and even correlations between values in different attributes. Several recent works on schema matching are based on *combining* multiple techniques in a principled fashion [Madhavan *et al,* 2001; Doan *et al,* 2001; Do and Rahm, 2002].

This paper accompanies an invited talk at the conference that surveys the successes and challenges in data integration to date.

Corpus-based matching

Conceivably, detailed knowledge about the domain in which the matching is being performed can be an important resource in a schema matching system. However, creating an appropriate KB is often hard, and furthermore, the result may be brittle in the sense it only helps on its domain of coverage, and only provides a *single* perspective on the domain.

We are pursuing an alternate approach in which knowledge is gleaned by analyzing a large corpus of database schemas and previously validated mappings. There are two types of knowledge that we can glean from such a corpus. First, we can learn the different ways in which words (or terms) are used in database structures (i.e., as relation names, attribute names and data values). Second, the validated mappings show how variations in term usages correspond to each other in disparate structures.

Although such a corpus is not easy to construct, it is a very different kind of activity than building a detailed and comprehensive knowledge base. It does not require the careful ontological design as a knowledge base does, nor the careful control on its contents, thereby removing the key bottlenecks present in the design of knowledge bases. The corpus offers *multiple perspectives* on modeling a particular domain, including different coverages of the domain. Thereby, it is more likely to provide knowledge that is useful for matching two disparate schemas.

In our initial work on the LSD system [Doan *et al,* 2001], we investigated the benefit of learning from previously validated mappings. In [Doan *et al,* 2001] we considered the case where multiple data sources are mapped to a single *mediated schema,* on which users pose queries. We provided LSD with the mediated schema and a set of *training matches* for some data sources. LSD used these matches to learn models of the elements of the mediated schema. Since no single learning algorithm captures all the cues from the domain, we used a multi-strategy approach that combined the predictions of several learners. We then asked LSD to predict matches between the mediated schema and a set of test schemas. Our experiments showed that (1) it is possible to achieve high accuracy with multi-strategy learning, and (2) additional accuracy is obtained by considering domain constraints (i.e., a simple form of domain knowledge). Overall, LSD achieved matching accuracy of 75-90% on small to medium sized schemas of data sources on the Web. We extended LSD to consider simple taxonomies of concepts in [Doan *et al,* 2002].

In recent work [Madhavan *et al,* 2003], we investigate the benefit of a corpus of schemas and matches, and the ability to use such a corpus to predict mappings between a *pair* of schemas that have not been previously seen. Like in LSD, we learn models for elements in the corpus, using both the information available in the schema and validated matches that are provided in the corpus. Given two schemas, *S1* and S2 we calculate for each element in them a *similarity vector* w.r.t. the corpus, i.e., how similar each element in S_i is to each element in the corpus. Very roughly speaking, if the similarity vectors of two elements *a\ E S1* and *a2 € S2* are similar to each other, then we predict that *a1* matches a2. The results of our experiments show that (1) even with a modest corpus of 10 schemas we are able to achieve good accuracy, and (2) the correct matches found by using the corpus and those found by other previously known techniques overlap, but have significant differences. Hence, the use of the corpus is finding matches that would not have been predicted by other techniques.

3 The corpus-based representation principle

This paper argues that using a corpus of domain models is a general technique that can be applied in many applications as an alternative or complement to traditional knowledge representation. Section 3.1 describes a few such applications, and Section 3.2 presents the components of a corpus-based representation system.

3.1 Applications of corpus-based representation

The following are classes of applications that can benefit from corpus-based representation.

Web search and query answering: the first class of applications concerns various information finding tasks on the Web (or intranet). The first application is *query answering* on the web (e.g., [Kwok *et al,* 2001; Radev *et al,* 2002]): a natural language query is posed to a web search interface, and rather than finding relevant pages, the search engine tries to find the answers to the query. A second, very related application, is *focused crawling* (e.g., [Sizov *et al,* 2003]), where the search engine is given a particular topic, and it tries to find pages relevant to it by starting at a set of initial pages, and crawling from them in a focused fashion. In both of these applications, the presence of additional domain knowledge has been shown or argued to be useful. However, the cost of constructing knowledge bases with such wide domain coverage would be prohibitive.

Creating and querying structured knowledge: one of the greatest impediments to using database and knowledge base technology is the conceptual difficulty of dealing with structured data versus unstructured text (the so called *structure chasm* [Halevy *et al,* 2003a]). It is well known that creating domain models (whether for database schema or knowledge bases) is a conceptual effort. Querying requires a different kind of effort - understanding someone else's domain model. Hence, a significant challenge for the KR and Database communities is to create tools that facilitate the creation and querying of structured data. [Halevy *et al,* 2003a] argues that a corpus of schemas enables building several tools for this purpose.

One such tool would be a *schema design advisor,* which assists in the authoring of structured data, much in the spirit of an auto-complete feature. A user of the such a tool creates a schema fragment and some data in a particular domain, and the tool then proposes extensions to the schema using the corpus. The user may choose a schema from the list and modify it further to fit the local context. Note that besides time savings, such a tool has other advantages, such as possibly resulting in better designs and helping users conform to

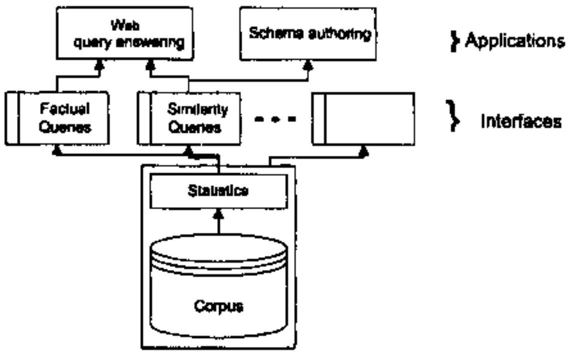

Figure 1: A corpus-based representation system contains a large collection of disparate domain models, instance data, validated mappings between domain models and meta-data. Applications access the corpus and the statistics computed on it through a set of interfaces. The interfaces support different types factual queries and similarity queries.

certain standards, when these are applicable. One can also imagine using corpus-based representation during the phase of verifying the correctness of a knowledge base. Hence, corpus-based representation can be an aid in creating traditional knowledge-based systems.

On the querying side, corpus-based representation can facilitate the querying of unfamiliar data. Specifically, consider a tool that enables you to pose a query using *your own terminology* to any database. The tool would then use the corpus to propose reformulations of the your query that are well formed w.r.t. the schemas of the database at hand. The tool may propose a few such queries (possibly with example answers), and let you choose among them or refine them.

General properties of candidate applications: the exact characterization of which applications lend themselves to corpus-based representation is a subject for future research. One property of such applications that immediately stands out is that they use the knowledge base as an aid in resolving ambiguity. That is, the applications use a set of techniques to propose plausible answers to a task, and then use the knowledge base to help *rank* the answers, *rule out* certain ones, or *guide* the search for answers. Hence, beyond the applications mentioned above, corpus-based representation can be useful in learning tasks, natural language processing and interfaces [Popescu *et al,* 2003], and information retrieval. In fact, as we explain in Section 4, corpus-based representation was inspired by the use of corpora (of different kinds!) in natural language processing and information retrieval. In contrast, corpus-based representation would be of limited use in applications that require intricate logical inference on carefully designed domain models (e.g., medical diagnosis, control of spacecraft, or reasoning about income-tax rules).

3.2 Corpus-based representation systems

We now offer some specific details on how corpus-based representation systems can be built and used. We describe the types of contents we may put in a corpus, and the interfaces that the corpus provides to an application (see Figure 1).

Contents of a corpus

A corpus can include any kind of information related to structured data. In particular, it includes the following.

1. Various forms of domain-model information: on the less expressive end, the corpus can include relational and object-oriented database schema or entity/relationship diagrams, XML DTDs or schemas, possibly with associated integrity constraints (e.g., functional dependencies). On the more expressive end, the corpus can include terminologies.

2. Instance data: we can include the actual rows of tables (or representative rows), XML documents, or the assertion (A-Box) component of a terminological knowledge base or Horn theory. In fact we can include data sets that do not have a schema (e.g., certain file formats). Note that it is very often the case that elements in the schema of one model may be instance data in another model. Hence, the distinction between schema and instance data is not clean cut.

3. Validated mappings: mappings can be given directly between pairs of domain models, or go through an intermediate domain model [Madhavan *et al.,* 2002].

4. Queries: queries (posed by users or applications) provide important information about how certain data is used. For example, when a database query performs a join over attributes of two different tables, that indicates that the columns are modeling the same domain (and this often not evident from the schema that only specifies the data type).

5. Other meta-data: there are many forms of meta-data that accompany domain models. They range from text descriptions of the meaning of different fields to statistics about table cardinalities or histograms on the set of possible values within a column.

It is important to emphasize that a corpus is *not* a logically coherent universal database. Rather, it is a collection of disparate structures, and there is relatively little control on the logical design of elements of the corpus. We expect that the domain-model information of the corpus will be stored and accessed using tools for model management [Bernstein, 2003], which attempt to provide a set of operators for manipulating models of data (as opposed to the data itself).

Statistics on the corpus

There is a plethora of possible analyses that can be performed on such a corpus, and finding the most effective ones is a long term research challenge. Below we describe certain kinds of statistics that can be computed over the corpus. We classify them according to whether they apply to individual words or terms, partial structures, or to elements of particular schemas.

Word and term statistics: these statistics are associated with individual words (in any language) and with noun or verb phrases. These statistics indicate how a word is used in different roles in structured data. For each of these statistics, we can maintain different versions, depending on whether we take into consideration only word stemming, synonym tables, inter-language dictionaries, or any combination of these three. The statistics include:

1. Term usage: How frequently the term is used as a relation name, attribute name, or in data (as a percent of all of its uses or as a percent of structures in the corpus).

2. Co-occurring schema elements: For each of the different uses of a term, which relation names and attributes tend to appear with it? What tend to be the names of related tables and their attribute names? What tend to be the join predicates on pairs of tables? Are there clusters of attribute names that appear in conjunction? Are there mutually exclusive uses of attribute names?

3. Similar names: For each of the uses of a term, which other words tend to be used with similar statistical characteristics?

Composite statistics: the same statistics can be applied to *partial structures.* Examples of partial structures are sets of data instances, relations with associated attribute names, a relation with some data (possibly with missing values).

Clearly, we need to significantly limit the number of partial structures for which we keep statistics (e.g., use techniques for discovering partial structures that occur frequently e.g., [Polyzotis and Garofalkis, 2002]). Given statistics for certain partial structures, we can estimate the statistics for other related structures.

Statistics for schema elements: the same word, used in different structures, can have different meanings. Hence, we may want to characterize the specific usages of terms in structures, and relate them to usage of terms in other structures. For example, in [Madhavan *et al.,* 2003] we learn a classifier for every relation and attribute name in the corpus. Following [Doan *et al,* 2001], we use meta-strategy learning. The training data used for learning is gleaned from the schema to which the element belongs and the training data of elements that have been mapped to it by a validated mapping in the corpus. Intuitively, the classifier is meant to recognize the particular usage of the term, even if it appears differently in another structure.

Application interfaces to the corpus

Applications built on a corpus should be able to access directly the statistics computed on the corpus, and perform the obvious lookups and queries on the corpus. In addition, just as knowledge based systems provide a set of higher-level interfaces (e.g., in the spirit of [Chaudhri *et al.,* 1998]), the same could be done for corpus-based representation systems. The following are examples of functions such an interface could support. Note that the implementation of these interfaces poses many research challenges.

We classify the interfaces on whether they ask for *factual* queries, or *similarity* queries. In all cases, we expect that the answers returned will be a ranked list of answers. In addition, as in traditional knowledge bases, we should be able to obtain an *explanation* for the answers. In the notation used below, variables are prefaced by $.

Factual information: in the simplest case, a query is a non-ground formula, and the corpus should return the possible groundings of the formula. As a simple example, we can ask for the values of $x, where Killed($x, Elvis). A different example would be a higher-order query, e.g., find all the $P, such that $P(Toyota, Honda).

A more complicated function would take as input a set of constants and return formulas that include all of these constants. Here, the goal is to find how two terms are related without specifying exactly what role they play in a formula. For example, the input may be *GPA* and student ID, and the answers could be GPA(studentID, $value) and Student(studentID, GPA, address). Note that the two terms play different roles in the two answers, and that the answers are templates for formulas (i.e., schema description), rather than particular sets of formulas.

In both of these cases, one can also complement the query with background knowledge. The knowledge may eliminate some answers or result in a different ranking of answers.

Similarity information: one of the key imports of the corpus is that it contains *different* ways of saying similar facts. Similarity queries try to get at these different perspectives. In the simplest case, a query can be a ground fact, and the corpus will return similar ways of saying the same thing. For example, the input can be Class(Lexus, luxury), and the output could be LuxuryCar(Lexus, Toyota), or even Car-Review(Lexus, LuxuryClass, VeryGood, goodTires).

In a more complex case, the query would consist of two terms, and the answer would be sets of facts in which these terms tend to play similar roles. For example, the input could be Review and Referee, and the answer would be IjcaiReview(paper1, reviewer2, accept) and A|Journal-Referees(round2, paper3, reviewed, reject). The answer could also specify where the terms are used differently, such as refereeing a soccer game and reviewing a movie.

In summary, we note some of the fundamental challenges in building an interface to a corpus. First, unlike logical knowledge/databases, answers are ranked rather than defining a boolean condition on data. Second, we are often interested in similarity querying, looking for different ways of saying the same thing [2]. Third, the answers returned often summarize *sets* of facts or describe schema fragments. Finally, it is often useful to tell the user what is *not* an answer.

4 Related work

The corpus-based representation approach stands in stark contrast to an approach like the Cyc Project [Lenat and Guha, 1990] that attempts to build a comprehensive knowledge base of all common sense. A corpus is a collection of any set of small domain models that are not centrally designed or coordinated. Importantly, the corpus is likely to include several models of the same domain. One of the key engineering challenges that arose in building Cyc was how to enable several knowledge engineers to add to such a large knowledge base without having to understand the entire knowledge base. *Contexts* (e.g., [Guha, 1991; Nayak, 1994; Giunchiglia and Ghidini, 1998]) are a mechanism proposed to address this complexity, where each knowledge engineer can focus on a small knowledge base, and then a set of lifting rules makes sure the entire knowledge base is semantically related. Unlike a corpus, contexts are still logically coordinated through a set of lifting rules.

The use of a corpus of structured data was first proposed in [Halevy *et al.,* 2003a]. There, the goal was to create tools that facilitate the creation, querying and sharing of structured data. That work was partially inspired by the use of large corpora in information retrieval and in statistical natural language processing (e.g. [Charniak, 1997J).

Several works have considered creating knowledge bases by a large set of independent contributors. The Open Minds Project [Singh *et al.,* 2002] collects knowledge from contributors on the web and builds a KB of common sense. Although the contributes are independent, the knowledge entry is guided by a set of templates, and hence the creation of a coherent KB is possible. [Richardson and Domingos, 2003] describe how to create a feedback loop by which consumers of knowledge provide indirect feedback to contributors of knowledge as to the usefulness of their contributions, thereby ultimately filtering out the bad contributions.

5 Conclusions

The analysis of large corpora is the key to the success of Information Retrieval techniques and to recent progress on Natural Language Processing. Corpus-based representation is predicated on the idea that large corpora of domain models can be leveraged to create novel tools in service of knowledge representation. The key advantage of corpus-based representation is that it avoids the need for careful logical design of a single comprehensive ontology. In addition, since a corpus will contain multiple ways of representing the same information, it opens up opportunities for exploring transformations on data and addressing problems related to representational heterogeneity. This paper laid the foundation for corpus-based representation, by describing preliminary versions of the contents of a corpus, the interfaces to it, some applications in which it would be a powerful representational tool, and an application from which the approach was inspired.

Corpus-based representation offers many exciting research challenges, not the least of which is actually collecting a large enough corpus to be of interest. In addition, there is a question about how focused the corpus needs to be in order to be useful in a particular domain. That is, can it only have domain models in that domain? Can domains models in other (possibly related) domains be useful, or do they only introduce noise that degrades the performance? Finally, no matter how a corpus is constructed, it can probably be manually *tuned* to perform even better. What form does such tuning take? While the challenges for corpus-based representation are enormous, we believe the payoffs could be huge, and the results can profoundly impact how we create and use structured knowledge.

Acknowledgments

Many colleagues have contributed to the ideas that led to corpus-based representation. We would like to thank Phil Bernstein, Anhai Doan, Pedro Domingos, Oren Etzioni, Zack Ives, Pradeep Shenoy, and Igor Tatarinov. Funding was provided by an NSF CAREER/PECASE Grant 11S-9985114, ITR Grant IIS-0205635, a Sloan Fellowship, and gifts from Microsoft Research Intel, NEC and NTT.

[2]There is a large literature on similarity querying in databases and content based image retrieval, but these focus on similarities between objects represented as points in multi-dimensional space.

References

[Bernstein *et al,* 2002] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing : A vision. In *Proceedings of the WebDB Workshop,* 2002.

[Bernstein, 2003] Philip A. Bernstein. Applying Model Management to Classical Meta Data Problems. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR),* 2003.

[Chamiak, 1997] Eugene Chamiak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings ofAAAI,* pages 598-603,1997.

[Chaudhri *et ai,* 1998] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James Rice. OKBC: a programmatic foundation for knowledge base interoperability. In *Proceedings ofAAAI,* pages 600-607,1998.

[Do and Rahm, 2002] Hong-Hai Do and Erhard Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proc. of VLDB,* 2002.

[Doan *et al,* 2001] Anhai Doan, Pedro Domingos, and Alon Halevy. Reconciling schemas of disparate data sources: a machine learning approach. In *Proc. ofSIGMOD,* 2001.

[Doan *et ai,* 2002] Anhai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *Proc. of the Int. WWW Conf,* 2002.

[Drapersa/.,2001] Denise Draper, Alon Y. Halevy, and Daniel S. Weld. The nimble integration system. In *Proc. ofSIGMOD,200\.*

[Giunchiglia and Ghidini, 1998] Fausto Giunchiglia and Chiara Ghidini. Local models semantics, or contextual reasoning = locality + compatibility. In *Proceedings of KR,* pages 282-291,1998.

[Guha, 1991] Ramanathan V. Guha. *Contexts: A Formalization and Some Applications.* PhD thesis, Stanford University, Stanford, C A, 1991.

[Halevy *et ai,* 2003a] Alon Halevy, Oren Etzioni, Anhai Doan, Zachary Ives, Jayant Madhavan, Luke McDowell, and Igor Tatarinov. Crossing the structure chasm. In *Proceedings of the First Biennian Conference on Innovative Data Systems Research (CIDR),* 2003.

[Halevy *et ai,* 2003b] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *Proc. of ICDE,* 2003.

[Halevy, 2001] Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal,* 10(4), 2001.

[Kalnis *et ai,* 2002] P. Kalnis, W. Ng, B. Ooi, D. Papadias, and K. Tan. An adaptive peer-to-peer network for distributed caching of olap results. In *Proc. of SIGMOD,* 2002.

[Kwok *et al,* 2001] Cody Kwok, Oren Etzioni, and Dan Weld. Scaling question answering to the web. In *Proc. ofthelnt. WWWConf,pages* 150-161,2001.

[Lenat and Guha, 1990] Douglas B. Lenat and R.V. Guha. *Building Large Knowledge Bases.* Addison Wesley, Reading Mass., 1990.

[Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS,* 2002.

[Madhavan *et al,* 2001] Jayant Madhavan, Phil Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *Proceedings of the International Conference on Very Large Databases (VLDB),* 2001.

[Madhavan *et ai,* 2002] Jayant Madhavan, Philip Bernstein, Pedro Domingos, and Alon Halevy. Representing and reasoning about mappings between domain models. In *Proceedings ofAAAI,* 2002.

[Madhavan *et ai,* 2003] Jayant Madhavan, Philip Bernstein, Kuang Chen, Alon Halevy, and Pradeep Shenoy. Matching schemas by learning from others. In *Working notes of the IJCAI-03 workshop on Data Integration on the Web,* 2003.

[MQSERIES, 2003] http://www-3. ibm.com/software/ts/mqseries/, 2003.

[Nayak, 1994] P. Pandurang Nayak. Representing multiple theories. In *Proceedings of AAAI,* pages 1154-1160,1994.

[Noy and Musen, 2002] Natalya Noy and Mark A. Musen. PROMPTDIFF: A fixed-point algorithm for comparing ontology versions. In *Proceedings ofAAAI,* 2002.

[Polyzotis and Garofalkis, 2002] Neoklis Polyzotis and Minos N. Garofalkis. Statistical synopses for graph-structured XML databases. In *SIGMOD '02,* 2002.

[Popescu et al.,2003] Ana-Marie Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the conference on Intelligent User Interfaces,* 2003.

[Radev *et ai,* 2002] Dragomir R. Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. Probabilistic question answering on the web. In *Proc. of the Int. WWWConf*, pagesz 408-419,2002.

[Rahm and Bernstein, 2001] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal,* 10(4):334-350,2001.

[Richardson and Domingos, 2003] Mathew Richardson and Pedro Domingos. Building large knowledge bases by mass collaboration. Technical Report 03-02-04, Department of CSE, University of Washington, Seattle, WA, 2003.

[Singh *et ai,* 2002] P. Singh, L. Thomas, E. Mueller, G. Lim, T. Perkins, and W. zhu. Open mind common sense: Knowledge acquisition from the general public. In *Proceedings of ODBASE,* 2002.

[Sizove/a/.,2003] S. Sizov, M. Biwer, J. Graupmann, S. Siersdorfer, M. Theobald, G. Weikuzm, and P. Zimmer. The BINGO! system for information portal generation and expert web search. In *Proceedings of the First Biannual Conference on Innovative Data Systems Research (CIDR), Asilomar(CA),* 2003.