Probabilistic Methods For Querying Global Information Systems

Nilesh Dalvi,
Department of Computer Science and Engineering,
University of Washington

July 14, 2004

Abstract

As databases undergo a paradigm shift from stand-alone applications to an architecture that supports data sharing on a global scale, they have to increasingly cope up with a fundamental problem: that of handling uncertainty and lack of information. Users often have to formulate queries over unfamiliar data sources: quite often they don't understand their schema, or their particular representation of data values.

Probabilistic methods have been used for a long time in the AI community to model uncertainties in knowledge bases, but a similar treatment is missing in databases and database queries. The goal of this report is to propose the use of queries on probabilistic databases to cope with uncertainties resulting from the schema and data heterogeneity in global data sharing and exchange.

1 Introduction

During the last few years, as information systems have become increasingly distributed and 'global', the challenges of data integration have increased. The volume of data, the variety of data formats, and users' velocity requirements for access to information have all increased dramatically.

A study [7] has shown that the volume of structured data on the web is around 500 times larger than the unstructured data that search engines index. Increasing magnitudes of content-rich databases from universities, libraries, associations, government agencies and businesses around the world is becoming publicly available. The freedom of information act passed by U.S. legislation requires data underlying all federally-funded research projects to be published upon request [4]. Data sharing is being made a condition for publication by scientific journals [28, 38] and funding agencies [39].

A prime challenge in integrating the data is the presence of *heterogeneity*. The data sources have different data models, schemas and data representations. They vary in the quality and reliability of their content, and are often inconsistent. Hence, data integration systems have to deal with *uncertain* and *incomplete information*.

The approach taken by databases towards queries is inadequate to deal with uncertainties. While database queries have a rich structure and a precise semantics, making it possible for users to formulate complex queries, it requires the users to have a detailed knowledge and understanding of data and its structure. This is seldom true in a data integration setting.

The Information Retrieval community, on the other hand, has taken a philosophically different approach to queries to meet these challenges. A query in Information Retrieval (IR) is just a set of keywords and is easy to formulate. IR queries offer two important features that are missing in

databases: the results are *ranked* and the matches may be *uncertain*, i.e. the answer may include documents that do not match all the keywords in the query.

Applications that retrieve data from global information sources have diverse needs. Some applications like managing bank accounts or airline booking require high *precision* while some applications, like searching for used vehicles or library catalogs, prefer high recall over accuracy. To support these broad spectrum of applications, a data integration system needs to understand the uncertainties that are inherent in it and their implications on query answers. Thus, there is a need for a framework for data integration systems to manage uncertain information.

Probabilistic methods have been used for a long time in the AI community to model uncertainties in knowledge bases. However, relatively little work has been done on uncertainties in databases. In this thesis proposal, I show that a data integration system can provide better functionality with a query-semantics based on relevances. Towards this end, I propose the use of a probabilistic framework to solve the range of problem arising in global data sharing.

Organization: This report is organized as follows. In section 2, I provide a survey of the sources of uncertainties that arise in data integration using examples of specific data integration scenarios. In section 3, I describe existing techniques for reasoning with uncertain and incomplete knowledge. In section 4, I identify the main challenges in managing uncertain information in data integration. In section 5, I conclude with a list of relevant research problems and deliverables for the thesis.

2 Uncertainties in Data Integration

In this section, I motivate the problem by giving examples of data integration scenarios that require reasoning about uncertain or incomplete information.

2.1 Queries over Unfamiliar Databases

Many data integration systems use a *mediator* architecture to provide uniform information access across heterogeneous information sources. Mediators provide a global schema to the users and then translate user queries from the global schema to individual source schemas.

Figure 1 describes a book-search mediator over two online bookstores, Amazon(`www.amazon.com`) and BN(`www.barnesandnoble.com`). The mediator provides the users with a global schema consisting of two tables, `author` and `category`. Users formulate their queries over this schema and are translated to the individual sources by mediator for execution.

Consider a user who is interested in a list of authors who have written books on '`Machine Learning`'. She poses the following query to the book-search mediator:

$$\varphi\texttt{(auth) := author(Author,Book), category(Book,'Machine Learning')}$$

One of the fundamental problems the user has to face is her lack of knowledge of how various sources represent book categories. The above query returns no answers as none of the databases have a category '`Machine Learning`'.

Amazon database has a category '`CS`→`AI`→`Machine Learning`' that corresponds to the query but does not match verbatim. The *BN* database does not have a corresponding category but it has related categories: '`Data Mining`', '`Robotics`' and '`Algorithms`'.

Thus, mediator has to solve the difficult problem of determining if a tuple in a arbitrary database satisfies the predicate in user query. There are several techniques it can employ to match for this purpose including edit distances, ontology-based distances [42], IDF-similarity and QF-similarity [3]

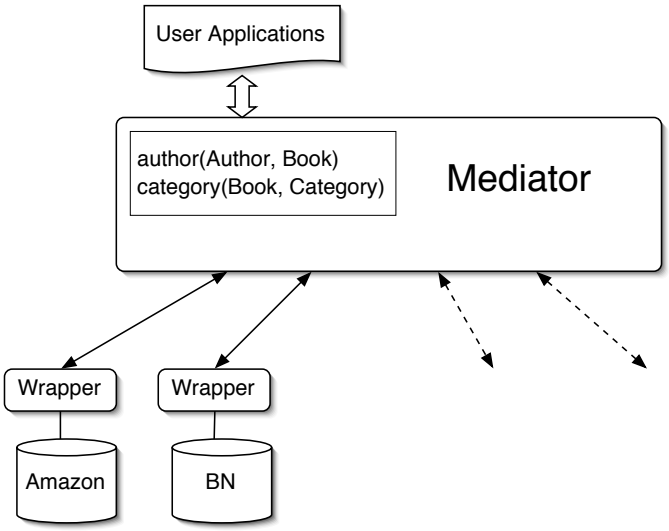

Figure 1: A Book-search Mediator

and lexicons like Wordnet [1]. However, the techniques only serve as evidences for matches and give the mediator degrees of confidences [1].

Thus, a mechanism is needed for translating the confidences in these matches into confidences in answering queries that can give relevance to query answers.

2.2 Data Mappings

Often, a data integration system has to combine information from multiple sources that talk about same objects. The differences in data representation leads to the problem of *identity uncertainty*. Identity uncertainty arises when objects are not labeled with unique identifiers globally or when these identifiers cannot be perceived. For instance, different sources refer to the same person by "John Doe", "Doe, John", "J. Doe" etc. It is a pervasive problem in real-world data analysis and is studies independently in different contexts, e.g. determining whether two citations refer to same publication, called *citation matching* [41], and whether two tuples from different databases refer to same entity, called *record linkage* [23].

To search data in such environments, people have proposed [33, 27] the use of mapping tables that store the correspondence between values. Such mappings are widely used in biological domains [14].

As an example, consider a data integration problem with two bookstores, S_1 and S_2. S_1 contains information about books and authors, while S_2 stores books and categories. Assume that the mappings are specified using *local-as-view (LAV)*, i.e. each data source is represented as a view over the global schema. We show the view definitions when all data sources use the same identifier for books (Ideal World Scenario) and how to represent mapping tables when each data source chooses a different book representation (Real World Scenario).

Ideal World Scenario: In this case, the global schema consists of a table $T(B, A, C)$ containing tuples of books, authors and categories. The sources are defined by views as follows:

[1] In the terminology for knowledge bases, it is called *degrees of beliefs*

$$
\begin{aligned}
S_1(B, A) &:= T(B, A, C) \\
S_2(B, C) &:= T(B, A, C)
\end{aligned}
$$

Common World Scenario: Here, the global schema consists of a table $T(B, A, C)$ along with two tables $M_1(B_1, B)$ and $M_2(B_2, B)$. Table $M_1(B_1, B)$ represents how a book symbol B_1 that appear in the source S_1 maps to the actual book entity B. M_2 represents the corresponding map for S_2. The view definitions for the sources now become:

$$
\begin{aligned}
S_1(B_1, A) &:= T(B, A, C), M_1(B_1, B) \\
S_2(B_2, C) &:= T(B, A, C), M_2(B_2, B)
\end{aligned}
$$

Mapping table represents the following view:

$$
M(B_1, B_2) := M_1(B_1, B), M_2(B_2, B)
$$

There are two important points to make note of. First, although mapping tables are viewed traditionally as materialized tables, they can also be considered as an abstraction to think about the problem of identity uncertainty. For instance, in the above example, $M(B_1, B_2)$ can also be used to represent an algorithm that, given two books, returns their similarity. Second, mapping tables have been used traditionally only as lookup tables to translate queries from one source to another. For instance, given a book in source S_1, the mapping table can be used to find information about the same book in S_2. However, by using view definitions described about, all the queries that can be posed in 'Ideal World' can now be posed in 'Common World', as users can now ask arbitrary queries on the table $T(B, A, C)$.

It is been suggested [27] that mapping tables should not only store the associations but also the confidence in this associations. In some domains, mapping tables are created by domain specialists as they require expert knowledge. Domain experts have varying level of expertise and therefore, need a mechanism for specifying their confidence in the mappings. Often, the process has to be automated because of the ubiquitous need for mappings and the sheer volume of data that needs to be integrated. Numerous algorithms [11, 41] for identity matching exist and use techniques ranging from simple string matching to Relational Probability Models and Markov Chain Monte Carlo inferencing. The algorithms ascertain matches with probabilities associated with them.

Thus, techniques are required for answering queries using views when there are probabilities associated with tuples in the views.

Let us see the implications of mapping tables on query rewritings. Consider a query that asks for a list of authors who have written a book on databases. It can be formulated as the following query on the global schema:

$$
\varphi(A) := T(A, B, \texttt{databases})
$$

Under Ideal World Scenario, the query can be answered using the following rewriting:

$$
\varphi'(A) := S_1(B, A), S_2(B, \texttt{databases})
$$

This query plan gives all the *certain answers* to the user query, i.e. answers that hold in all the databases consistent with the views.

However, the same query, in Real World Scenario, has *no certain answers* under open-world assumption. It can be observed that the rewriting below does not return certain answers:

$$\varphi''(A) : S_1(B_1, A), S_2(B_2, \texttt{databases}), M(B_1, B_2)$$

To see this, consider three tuples $S_1(b_1, a)$, $S_2(b_2, \texttt{databases})$ and $M(b_1, b_2)$. The three tuples lead to a as an answer to the query. However, the views may represent the following database: $T(b, a, \texttt{AI})$, $T(b', a', \texttt{databases})$, $M_1(b', b_1)$, $M_1(b, b_1)$, $M_2(b', b_2)$.

In general, an object in one source can get mapped to several objects in another source (with various probabilities). While there are no certain answers, probabilistic methods are required to return meaningful answers.

2.3 Database Consistency

Sometimes certain integrity constraints are believed to hold in the global system. For instance, every person has a unique address and every paper is published in a unique conference. These integrity constraints are used in the generation of the query plan. There are situations where without integrity constraints no query plan can be generated [17, 22].

These global constraints are often violated due to inconsistencies in data sources. Even in the presence of consistent data sources, a global system that integrates them may become inconsistent. A taxpayer database and a voter registration database may have conflicting addresses for the same person. Each of those databases separately satisfies the functional dependency that associates a single address with each person, and yet together they violate this dependency.

As an illustrative example, consider a data integration scenario where the global schema consists of the following relations:

$$\begin{aligned} &\text{affiliation(Person, Organization)} \\ &\text{location(Organization, Location)} \end{aligned}$$

The first relation describes the organization where a person works and the second relation describes the locations of organizations. A person only works in a single organization and each organization has a single location. Therefore, we have the following constraints on the global schema

$$\begin{aligned} \text{affiliation} &: \text{Person} \rightarrow \text{Organization} \\ \text{location} &: \text{Organization} \rightarrow \text{Location} \end{aligned}$$

Suppose we have the following data sources:

$$\begin{aligned} s_1(P, O) &:- \text{affiliation}(P, O) \\ s_2(P, L) &:- \text{affiliation}(P, O), \text{locations}(O, L) \end{aligned}$$

s_1 stores part of affiliation table while s_2 stores, for each employee, its work location. Assume a user wants to know the location of the organization '`UW`':

$$\varphi(L) \quad := \quad \text{location}(\texttt{`UW'}, L)$$

The following plan would answer it:

$$\varphi'(L) \quad := \quad s_1(P, \texttt{`UW'}), s_2(P, L)$$

If s_1 contains a tuple (`'person1', 'UW'`) and s_2 contains a tuple (`'person1', 'Seattle'`), we can infer that `'UW'` is in `'Seattle'`. Note that the integrity constraints on the global schema play a crucial role in the correctness of the above plan. Without the constraints, the query cannot be answered using the two sources.

Now suppose we access a third source s_3 with the same schema as s_2, i.e. it also stores the work locations of employees. However, it contains a tuple (`'person1', 'Google'`). This information is inconsistent with source s_2. Current semantics for query answering using views are not robust to dealing with inconsistencies. In this example, there cannot be any global database that satisfies the integrity constraints and is consistent with all the sources. Thus, no query could be answered.

Inconsistencies are a part and parcel of data integration. Often, sources update their data at different frequencies and hence contains outdated data. Many systems use automated tools to extract data from unstructured sources on the web and are prone to errors. For instance, systems like *Citeseer* automatically extract paper information from research papers. A major source of inconsistencies is the errors in schema matching. If some of the attributes of a source are incorrectly matched, all the data from that source could possibly be inconsistent with other sources.

Current approaches [37, 8, 32] to answering queries in inconsistent databases are based on the concept of a *repair*. They consider all minimal ways of repairing the database to restore its consistency. All of the resulting databases are considered as a possibility and they are used to define certain answers to the queries. As a result, only the tuples that correspond to the 'consistent part' of the database contribute to the query answers and all other tuples are winnowed out.

There are several reasons why this approach is not suitable for data integration. First, as we discussed earlier, inconsistencies are a rule rather than exception. Often a substantial fraction of the data participates in inconsistencies but still contains useful information. For instance if two sources return two different conferences for the same paper, there is a high chance that one of them is the actual conference. A technique that associates degrees of confidences to query answers can leverage this information. Another limitation of repair based approaches is that they ignore the fact that data sources differ in their quality and reliability. In practice, there are sources that are more trusted than others and schema matching algorithms have varying amount of confidences on the mappings they output on different sources.

2.4 Uncertainties in Schema Matching

Schema matching had traditionally been done by domain experts, which is a tedious, time-consuming and expensive approach and limited to environments where sources are small and stable. Scaling up data integration applications to a global scale requires automated tools for matching schemas and is an active area of current research. Several systems [34, 16, 40] for automated schema matching have been developed. A schema matching algorithm makes use of various basic similarity measures like 1) similarity in element names based on edit distances, N-grams and lexicons like Wordnet, 2) data types of various attributes in the schema, e.g. integers, strings and dates, 3) text descriptions or documentations, if available and 4) data associated with elements, e.g. addresses have a particular

format. These similarity measures are combined using techniques like rule-based logic and machine learning.

However, the similarity measures used by algorithms are only heuristics. Often element names are abbreviated or synonyms are used, text descriptions are not available, same data types have different representations and so on. Thus, matching algorithms produce mappings with various level of confidences.

Consider the following two schemas:

$$S_1(\texttt{PersonName, OfficeAddress, HomeAddress})$$
$$S_2(\texttt{name, r\_addr, o\_addr})$$

A schema matcher recognizes that `'PersonName'` maps to `'name'`. Both `'HomeAddress'` and `'OfficeAddress'` either map to `'r_addr'` or `'o_addr'`. Thus, it considers following two matchings:

$$M_1 \;:\; \texttt{PersonName} \rightarrow \texttt{name}, \texttt{OfficeAddress} \rightarrow \texttt{r\_addr}, \texttt{HomeAddress} \rightarrow \texttt{o\_addr}$$
$$M_2 \;:\; \texttt{PersonName} \rightarrow \texttt{name}, \texttt{OfficeAddress} \rightarrow \texttt{o\_addr}, \texttt{HomeAddress} \rightarrow \texttt{r\_addr}$$

The system places a higher confidence in M_2 because `o_addr` is possibly an abbreviation for `OfficeAddress`. Confidences can vary even across different parts of the same matching, e.g. `PersonName`→`name` is a certain match in both matchings.

More generally a schema matching algorithm over large schemas can produce several matchings with various degrees of confidence. An incorrect mapping can cause queries to return empty answers and lead to relevant tuples not being retrieved. Alternatively, it can result in incorrect answers and possible inconsistencies with other data sources. Thus, query answering algorithm need a tighter integration with the schema matching algorithms.

2.5 Incomplete Information

A problem closely related to uncertain matchings is that of incomplete matchings. Sources do not always contain enough information to answer a query exactly. Queries are posed over global schema and global schema contains the union of attributes of all sources. As a result, exact query translations are often not possible. For instance if one of the predicates in the query refers to the attribute *price*, it cannot be answered by a source that only contains price ranges or does not contain prices at all.

A sufficiently rich query may leave a large fraction of data sources untouched and consequently return very few or no answers. In such cases, to retrieve data from these sources, approximate query translations must be used. A case study [10] of a real-world data integration scenario shows that around 70% of query translations must rely on approximations to retrieve information and hence, are critical for real-world applications.

While an exact query can return no answer, a loose approximation can result in a lot of irrelevant answers. Thus, query answering requires calculations of the relevance of answers when queries use approximations.

The problem has been consider in limited settings. Chang et al. [10] use mapping tables that store the recall and precision with each mapping to generate the closest query translation using multiple closeness criteria like minimal-superset and maximal-subset. Florescu et al [20] consider the use of probabilistic knowledge in form of overlap between collections, overlap between information

sources and coverage of information sources. They address the problem of *ordering accesses* to the sources to maximize the likelihood of obtaining the answers as early as possible. However, current techniques do not consider the effect of approximations on the relevance of query answers.

Approximate query translation involves managing probabilistic schema mappings. These are fundamentally different from the uncertain schema mappings that we discussed in the previous section. The former involves statements like "*90% of books on 'Information Systems' in source S_1 are present under 'Databases' category in source S_2*". The latter involves statements like "*there is a 90% chance that all books of category 'Information Systems' in S_1 are under 'Databases' in source S_2*".

2.6 Query Secrecy

So far, we have looked at the problem of query answering. A different problem associated with global data sharing is the problem of *query secrecy*. The problem is defined as follows: given a set of views $V_1, V_2, \cdots$ that a source wants to publish over its data, do they logically disclose any information about a query S that it wants to keep secret? Miklau et al. [35] consider the problem of perfect query secrecy and consider it as a fundamentally different problem from query answering. Quantifying the amount of information that views leak about secret queries remains an open problem.

Surprisingly, the techniques used for answering queries with uncertainties also apply to measuring query secrecy as both involve reasoning with incomplete information. A query answer with complete certainty corresponds to perfect query answering and an answer with zero certainty corresponds to perfect query secrecy.

We illustrate the need for measuring the information leakage using following example. Consider a source that contains employee data with schema $employee(Name, Dept, Phone, Email)$ and following query secrecy problems:

$Problem1:$

$V_1(Name,\ Phone)\ :-\ employee(Name,\ 'CSE'\ ,\ Phone,\ Email)$

$S(Name,\ Phone)\ :-\ employee(Name,\ 'EE'\ ,\ Phone,\ Email)$

$Problem2:$

$V_2(Dept,\ Name)\ :-\ employee(Name,\ Dept\ ,\ Phone,\ Email)$

$V_3(Dept,\ Phone)\ :-\ employee(Name,\ Dept\ ,\ Phone,\ Email)$

$s(Name,\ Phone)\ :-\ employee(Name,\ Dept\ ,\ Phone,\ Email)$

$Problem3:$

$V_4(Name,\ Email)\ :-\ employee(Name,\ Dept\ ,\ Phone,\ Email)$

$S(Name,\ Phone)\ :-\ employee(Name,\ Dept\ ,\ Phone,\ Email)$

The query S is secret in problem 1, but not in problems 2 and 3. In problem 2, suppose a department has five people. Then, the two views separately provide a list of five names and five phone numbers in that department. An attacker can randomly guess the phone number of anyone with a 20% probability of success. In problem 3 the a priori probability that a random name and phone is an answer to S is extremely small. The probability can increase if the view V_4 reveals the name to be in the database. We see that amount of information gained about S is quite different in each case. Its zero for problem 1, very small for problem 3 and substantial for problem 2.

3 Existing Techniques for Handling Uncertainty

In this section, we describe the existing work on reasoning with uncertain knowledge and its relation to data integration.

3.1 Possible Worlds Semantics

To represent uncertainties arising in data integration, we saw the need for modeling two kinds of statements. The first kind deals with statements like "15% of all `CS` papers are `DB` papers". These statements represent statistical knowledge also known as *chance setups* in Knowledge Representation. It represents a chance setup where a randomly chosen `CS` paper has 0.15 probability of being a `DB` paper.

The second kind are statements like "the given paper has 0.15 probability of being a `DB` paper". This describes the confidence of the system on a statement, known as *degree of belief* in Knowledge Representation. The statement describes the existence of a number of possibilities, or *possible worlds*, in some of which the given paper is a `DB` paper. The first statement assumes that there is only one possible world, and in this world, there is some probability distribution over the set of `CS` papers.

The possible worlds semantics, originally put forward by Kripke [29] for modal logics, is commonly used for representing statements about degrees of beliefs. Halpern [25] has shown how First-Order logic can be extended to support probabilities with possible worlds semantics which we describe here. A first-order logic consists of a set or predicates, like `JournalPaper`(x) or `Teaches`(x,y), which are mappings from individuals to truth values. Statements in first-order logic are constructed using logical operators($\vee$,$\wedge$, $\neg$, $\exists$, $\forall$) and variable symbols[2] on the predicates. Under normal interpretation (non-probabilistic), there is a unique world that determine the contents of the predicates. Truth values of statement are decided using by a set of inferencing rules. Under possible worlds semantics, there is a set of worlds, each describing the contents of predicates. There is also a probability distribution over the worlds. The statements do not have a truth value but a probability associated with them. The probability of a statement is the sum of probabilities of all the worlds where the statement can be inferred.

Unfortunately, the number of possible worlds in exponential in the number of objects. Thus, to apply the possible world semantics to model uncertainty, efficient evaluation algorithms are required.

3.2 Probabilistic Databases

Probabilistic frameworks [9, 6, 15, 31] have been proposed in the past. Initial works made restrictive simplifying assumptions to get around the high complexity of query evaluation. Cavallo and Pittarelli [9] proposed a theory for probabilistic databases where they assume that tuples in the same relations represent disjoint events. Barbara et. al. [6] and Dey et al. [15] improved upon this model, but still allowed only a very restricted set of queries. The Probview system [31] used user defined functions to combine probabilities of events, but chose an alternate semantics for projection where duplicates are not removed and hence, do not assign a single probability to tuples.

Zimanyi [43] and Fuhr [21] used the logic of probabilities [25] to give semantics to queries in databases. They also give algorithms to evaluate queries in probabilistic databases based on an algebra of events. However, the algorithms run in time exponential in the size of the database and hence, impractical to use.

[2]There are also functional symbols but we ignore them in this discussion

In our recent work [13], we propose efficient algorithms for answering queries in probabilistic databases. We consider a database where every tuple has a certain probability of belonging to the database. The dependence between the probabilities is described by events. There is a set of basic events, each has a certain probability of occurrence, and each tuple has an event. Thus, if two tuples have the same event, either both of them are present in the data or none of them. All the basic events are assumed to be independent.

The resulting probabilistic database represents a probability distribution on the sets of possible subsets of the given database. Each such possible database represents a complex event that is the conjunction of the events of the tuples present in that instance. The probability of the complex events can be calculated from the probabilities of the basic events and using the independence assumption. Given any query, the answer is itself a set of tuples with probabilities. The probability of a tuple is the sum of probabilities of all possible database instances where that tuple in an answer to the query.

We show that certain queries have a #P-complete [3] data complexity under probabilistic semantics. However, a large fraction of queries that occur in practice do admit an exact efficient algorithm (8 out of the 10 TPC/H queries fall in this category). For others, we describe Monte-Carlo simulation algorithms to approximate the probabilities to arbitrary precision.

We also show how an imprecise query on a deterministic database implicitly defines a probabilistic database where the probabilities of the answers give their relevance.

3.3 Query Answering Using Views

Query answering using views is an instance of problem that requires reasoning with incomplete information. The query answering problem is the following: given a query Q over a schema and a set of view definitions over the same schema, how can Q be answered using only the answers to the views? The query answers are defined by the notion of *certain answers*, first introduced by Abiteboul and Duschka [2].

Let Q be a query and $V = \{V_1, \cdots V_m\}$ be a set of view definitions over a database schema R. Let $v = \{v_1, \cdots v_m\}$ be the extensions of the views. Two interpretations are consider for the view extensions. Under *closed-world model*, the view extensions are considered to be complete while under *open-world model*, the view extensions could be incomplete. In data integration settings, data sources are often defined as views over the global schema, called *local-as-view (LAV)* approach. The sources are assumed to be incomplete and hence, an open-world model is assumed.

A tuple a is called a certain answer to the query Q under open-world assumption given view extensions v if for all databases D that satisfy $v_i \subseteq V_i(D)$ for every i, we have $a \in Q(D)$.

A tuple a is called a certain answer to the query Q under closed-world assumption given view extensions v if for all databases D that satisfy $v_i = V_i(D)$ for every i, we have $a \in Q(D)$.

The certain answers have an interpretation under possible worlds semantics. The views do not define a unique database instance and hence there are several possible databases. However, the view extensions place certain constraints on the databases. The certain answers are precisely the tuples that have a probability 1 given any database that satisfies the constraints.

[3] The complexity class #P is the counting version of the class NP. #P-complete problems do not admit polynomial time solution unless $P = NP$

In section 2.2, we saw the need for answering queries over probabilistic views, i.e. when tuple membership in views is uncertain. In this setting, the current techniques for query answering do not help because there are no certain answers. Current techniques do not distinguish between *pretty certain* answers and answers that are certainly not true.

3.4 Ranking Queries in Databases

Most of the current research on ranking queries in databases has been non-probabilistic and devised for specific cases. The VAGUE system [36] supports queries with vague predicates, but the query semantics are ad hoc and apply only to a limited SQL fragments. Chaudhuri et al. [3] consider ranking query results automatically based on heuristics scores. Theobald and Weikum [42] describe a query language for XML that supports approximate matches with relevance ranking based on ontologies and semantic similarity. Keyword searches in databases are discussed in [26, 12, 24]. Fagin [19] gives an algorithm to rank objects based on its scores from multiple sources: this applies only to a single table.

4 Challenges in Managing Uncertainties in Data Integration

In previous sections, I have illustrated various sources of uncertainties in data integration and techniques used for representing and reasoning with knowledge. However, there are several challenges that must be met before these techniques can be applied to solve data integration problems.

Challenge 1: How can uncertain information be quantified?

The fundamental challenge is figuring out *what the uncertainties mean*? At elemental level, systems use various measures of closeness that include edit distances, q-grams, phonetic similarity, TF/IDF, ontologies and lexicons like Wordnet. But how can the similarity scores between data items be translated into probabilities of their matches?

Challenge 2: A system must manage a rich representation of uncertainties.

We saw that uncertainties arise at both schema level as well as data level. They involve statements about statistical knowledge as well as degrees of beliefs. The information can be uncertain, unreliable or incomplete. A system must be able to manage these various kinds of uncertainties.

Challenge 3: Probabilistic dependencies must be supported.

Answering queries involve combining multiple sources of uncertain information, and its affected to a great extent by the correlation between the uncertainties. *The data integration system must be able to understand and handle the dependence between probabilities.*

Challenge 4: A solution should scale to large number of sources and large amount of data.

Efficient algorithms are needed for query answering in presence of uncertainties. Techniques from Knowledge Reasoning are too impractical to be applied directly to data integration. The characteristics of data integration applications are 1) there is a large volume of data spread across a large number of sources and 2) queries are usually expressed as select/project/joins or conjunctive queries and their union, and thus use only a small fragment of logic.

5 Thesis Proposal

As a part of my thesis, I propose to develop techniques that can meet the challenges of managing uncertainties in data integration. Our initial work in this direction, on query evaluation in probabilistic databases [13], provides me a starting point for solving these challenges and I plan to build on this work. Following problems are of particular interest:

- **Generation of Probabilities:** The problem of generating probabilities from uncertainties has received very little attention in probabilistic databases, though it has been considered in other domains. For instance, Lakshmanan [30] has given an epistemic foundation for associating probabilities to agent beliefs in knowledge bases. Techniques are needed to learn probabilities using, for instance, machine learning tools on training data sets or user feedbacks.

  Another prospective direction is a query language that can explicitly talk about the uncertainties. Such a query language can be used by applications to specify their needs of high precision or recall, or to specify, for example, if certain predicates in the query are more important than others.

- **Support for dependencies between uncertainties:** In our current work or probabilistic databases, all the probabilities are assumed to be independent. As we have seen earlier, there are scenarios in data integration where this assumption does not hold. Supporting a language rich enough to express "interesting" probability dependencies is an important open problem. The trade-offs between the expressive power of the language and the complexity of query answering make the problem challenging as arbitrary dependencies can quickly lead to computational intractability.

  An important class of dependencies that occur very frequently in practice and demands attention is *mutual exclusion*. Two events are mutually exclusive if they both cannot happen together. Often it is the case that some data item is known to have a single unique value. This value, however, may be unknown and specified probabilistically. All the resulting probabilistic events are mutually exclusive. For example, every paper is published in a unique conference. Thus, while *conference(Paper1, SIGMOD)* and conference(Paper1, VLDB) both may have a finite probability, both of them cannot be true.

- **Integration of Uncertainties in Schema Matching:** We have seen that there is a close connection between the uncertainties in schema mappings and the relevance of the resulting answers. While schema matching algorithms do the hard work of calculating the probabilities of various matches, it is lost while answering queries. Usually, only the most probable candidate matching is considered as the output of the algorithm and is assumed to be the correct mapping.

  Schema matching algorithms can be more useful if they return a set of mappings along with the confidences or alternatively a single representation of the evidences and decision choices that schema matchers have. Thus, query answers across sources can be ordered by relevances, accesses to different sources can be ordered depending on queries and alternate mappings for a source can be tried that if its data mismatches with data from other sources with more certain mappings. In summary, the schema matching algorithms need a tighter integration with query answering.

- **Support for Statistical Knowledge:** While the tuple probabilities in our current model

represent confidences or degrees of beliefs, support is needed to represent statistical information.

The form of knowledge can vary from simple statistics like "15% of DB papers are AI papers" to statements like "the costs of products follow a Gaussian distribution". Statistical knowledge has been used to represent precision and recall of schema mappings [10] and to represent the overlap between collections and sources [20].

While degrees of beliefs define a probability distribution over possible worlds, statistical statements define a probability distribution over domain and effort is required to integrate both of them in a probabilistic database.

- **Semantics for query answering using views** There is need for a semantics for query answering on probabilistic views. In Section 3.3, we saw that probabilistic views have an interpretation as constraints on the probability distribution of database instances.

  Unfortunately, the set of constraints imposed by probabilistic views do not always define a unique probability distribution. Hence, possible world semantics cannot be directly applied to the problem. We describe here two different techniques in Knowledge Reasoning that can be employed.

  Bacchus, Grove, Halpern and Koller [5] consider the problem of generating new beliefs in a knowledge base from old. They call the beliefs *subjective information* and distinguish it from *objective information* or the facts in the knowledge base. They consider a prior knowledge about the probability distribution on the possible worlds. For instance, in the absence of any knowledge about the databases, all possible databases are equally likely (a uniform prior). The subjective information is treated as a set of constraints on the probability distribution on possible worlds. To obtain a unique posterior distribution, they consider the one "closest" to the prior distribution that satisfies all the constraints. They use the well studied measure of *cross-entropy* to determine the distance between two distributions.

  Fagin, Halpern and Megiddo describe a logic of probabilities [18]. They consider a language that allows statements like "the probability of E_1 is at least 0.8" and "the probability of E_2 is at least twice the probability of E_3". They give a inferencing algorithm for the language by considering it as statements over the probability distribution of possible worlds. Thus, when constraints on probabilities do not define a unique distribution, bounds can be imposed on the probabilities of various events. They apply techniques from linear programming.

  Thus, several questions need to be answered. Do both techniques result in equivalent semantics for answering queries and if not, which one is more meaningful for data integration? Also, how does this change the complexity of query answering?

- **Efficient evaluation algorithms:** In our previous work[13], we observed that the problem of evaluation of conjunctive queries on probabilistic databases is #-P complete. Prior works have suggested the use of a heuristic called *extensional evaluation*, where relational algebra operators calculate probabilities along the way. While extensional evaluation is fast and can be easily incorporated into a query execution engine, it does not give any guarantees on the errors and it also sensitive to the plan used for execution. Figure 2 shows the recall curve for the extensional evaluation of a TPC-H query and its very low. For instance, a user interested in top 50 answers gets a recall of only 40%.

  Our work revealed a surprising result: each query either has an execution plan on which extensional evaluation computes correct probabilities or its #-P complete to evaluate that

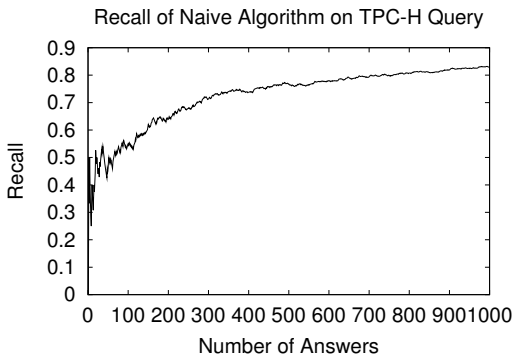

Figure 2: Recall Plot for Extensional Evaluation of TPC-H Query

query. We call them *safe* and *unsafe* queries respectively.

There are lots of questions that need to be answered. For safe queries, only some execution plans guarantee correctness. How can a query optimizer search over them to obtain a good plan? For unsafe queries, that is no efficient way to calculate the answers exactly (unless $P = NP$). Can query plans be produced that guarantee a small error? Alternatively, can techniques like Monte-Carlo simulations be employed efficiently to approximate the answers? We have only made initial progress in this direction.

The problem of efficient query evaluation would become progressively more challenging as we add the support for richer dependencies, schema uncertainties and statistical information.

Another problem of interest is the evaluation of top-k queries. Can the most relevant answers to a query be returned quickly without evaluating the query completely?

Fast algorithms for query answering are crucial for the success of using probabilistic methods in answering queries.

6 Conclusions

There is a pressing need for managing uncertainties in data integration systems. In this thesis proposal, I propose a probabilistic framework to represent and manage these uncertainties. I identified the various sources of uncertain information that arise in data integration, described the current techniques for reasoning with knowledge and the main challenges in applying these to data integration and put forward a set of specific problems that need to be solved.

References

[1] Wordnet 2.0: A lexical database for the english language: *http://www.cogsci.princeton.edu/ wn/*, Jul. 2003.

[2] Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263, 1998.

[3] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *Proceedings of the First Biennial Conf. on Innovative Data Systems Research*, 2003.

[4] Beth Azar. Information act opens data and debate. *American Psychological Association Monitor*, 30(8), September 1999.

[5] Fahiem Bacchus, Adam Grove, Joseph Halpern, and Daphne Koller. Generating new beliefs from old. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 37–45, 1994.

[6] Daniel Barbará, Hector Garcia-Molina, and Daryl Porter. The management of probabilistic data. *IEEE Trans. Knowl. Data Eng.*, 4(5):487–502, 1992.

[7] Michael K. Bergman. The deep web: Surfacing hidden value. 2001.

[8] L. Bertossi, J. Chomicki, A. Cortes, and C. Gutierrez. Consistent answers from integrated data sources, 2002.

[9] Roger Cavallo and Michael Pittarelli. The theory of probabilistic databases. In *VLDB'87, Proceedings of 13th Int. Conf. on Very Large Data Bases, September 1-4, 1987, Brighton, England*, pages 71–81, 1987.

[10] Chen-Chuan K. Chang and Hector Garcia-Molina. Approximate query translation across heterogeneous information sources. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 566–577, 2000.

[11] E. Charniak and R. P. Goldman. A bayesian model of plan recognition, 1993.

[12] S. Chaudhuri, G. Das, and V. Narasayya. Dbexplorer: A system for keyword search over relational databases. In *Proceedings of the 18th Int. Conf. on Data Engineering, San Jose, USA*, 2002.

[13] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. 2004.

[14] Susan Davidson, G. Christian Overton, and Peter Buneman. Challenges in integrating biological data sources. *Journal of Computational Biology*, 2(4):557–572, 1995.

[15] Debabrata Dey and Sumit Sarkar. A probabilistic relational model and algebra. *ACM Trans. Database Syst.*, 21(3):339–369, 1996.

[16] AnHai Doan, Pedro Domingos, and Alon Y. Levy. Learning source description for data integration. In *WebDB (Informal Proceedings)*, pages 81–86, 2000.

[17] Oliver M. Duschka, Michael R. Genesereth, and Alon Y. Levy. Recursive query plans for data integration. *Journal of Logic Programming*, 43(1):49–73, 2000.

[18] Ronald Fagin, Joseph Y. Halpern, and Nimrod Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1/2):78–128, 1990.

[19] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 102–113, 2001.

[20] Daniela Florescu, Daphne Koller, and Alon Y. Levy. Using probabilistic information in data integration. In *The VLDB Journal*, pages 216–225, 1997.

[21] Norbert Fuhr and Thomas Rolleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[22] J. Gryz. Query rewriting using views in the presence of functional and inclusion dependencies. 24(7):597–612, 1999.

[23] Lifang Gu, Rohan Baxter, Deanne Vickers, and Chris Rainsford. Record linkage: Current practice and future directions, 2003.

[24] Lin Guo, Feng Shao, Chavdar Botev, and Jayavel Shanmugasundaram. Xrank: Ranked keyword search over xml documents. In *Proceedings of the 2003 ACM SIGMOD Int. Conf. on Management of Data, San Diego, California, USA, June 9-12, 2003*, pages 16–27, 2003.

[25] Joseph Y. Halpern. An analysis of first-order logics of probability. In *Proceedings of IJCAI-89, 11th International Joint Conference on Artificial Intelligence*, pages 1375–1381, Detroit, US, 1989.

[26] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *Proc. 28th Int. Conf. Very Large Data Bases, VLDB*, 2002.

[27] Anastasios Kementsietsidis, Marcelo Arenas, and Rene J. Miller. Mapping data in peer-to-peer systems: semantics and algorithmic issues. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 325–336, 2003.

[28] Stephen H. Koslow. Should the neuroscience community make a paradigm shift to sharing primary data? *Nature Neuroscience*, 3(9):863–865, September 2000.

[29] S. A. Kripke. Semantic analysis of modal logic. i: Normal propositional calculi, 1963.

[30] Laks V. S. Lakshmanan. An epistemic foundation for logic programming with uncertainty. In *Foundations of Software Technology and Theoretical Computer Science*, pages 89–100, 1994.

[31] Laks V. S. Lakshmanan, Nicola Leone, Robert Ross, and V. S. Subrahmanian. Probview: a flexible probabilistic database system. *ACM Trans. Database Syst.*, 22(3):419–469, 1997.

[32] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Source inconsistency and incompleteness in data integration. In *Proc. of the 9th Int. Workshop on Knowledge Representation meets Databases (KRDB 2002)*, 2002.

[33] Bertram Ludscher, Amarnath Gupta, and Maryann E. Martone. Model-based mediation with domain maps. In *Proceedings of the 17th International Conference on Data Engineering*, pages 81–90, 2001.

[34] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.

[35] Gerome Miklau and Dan Suciu. A formal analysis of information disclosure in data exchange.

[36] Amihai Motro. Vague: a user interface to relational databases that permits vague queries. *ACM Trans. Inf. Syst.*, 6(3):187–214, 1988.

[37] Amihai Motro. Multiplex: A formal model for multidatabases and its implementation. In *Next Generation Information Technologies and Systems*, page 138, 1999.

[38] Nature Neuroscience Editorial. A debate over fmri data sharing. *Nature Neuroscience*, 3(9):845–846, Sep 2000.

[39] NIH statement on sharing research data. *http://grants2.nih.gov/grants/policy/data_sharing/index.htm*, March 2002. U.S. National Institutes of Health.

[40] D. S. Luigi Palopoli and D. Ursino. Semi-automatic semantic discovery of properties from database schemas. In *IDEAS*, pages 244–253, 1998.

[41] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching, 2002.

[42] Anja Theobald and Gerhard Weikum. The xxl search engine: ranked retrieval of xml data using indexes and ontologies. In *Proceedings of the 2002 ACM SIGMOD Int. Conf. on Management of data*, pages 615–615, 2002.

[43] E. Zimanyi. Query evaluation in probabilistic databases. *Theoretical Computer Science*, 171(1-2):179–219, 1997.