MYSTIQ: A system for finding more answers by using probabilities

Jihad Boulos*, Nilesh Dalvi†, Bhushan Mandhani†, Shobhit Mathur†, Chris Re†, Dan Suciu†

†University of Washington, USA. *American University of Beirut, Lebanon.

Mystic \Myst"ic\, n. Having an import not apparent to the senses nor obvious to the intelligence; beyond ordinary understanding.

– WordNet(R) Dictionary

1. MOTIVATION

MystiQ is a system that uses probabilistic query semantics [4] to find answers in large numbers of data sources of less than perfect quality. There are many reasons why the data originating from many different sources may be of poor quality, and therefore difficult to query: the same data item may have different representation in different sources; the schema alignments needed by a query system are imperfect and noisy; different sources may contain contradictory information, and, in particular, their combined data may violate some global integrity constraints; fuzzy matches between objects from different sources may return false positives or negatives. Even in such environment, users sometimes want to ask complex, structurally rich queries, using query constructs typically found in SQL queries: joins, subqueries, existential/universal quantifiers, aggregate and group-by queries: for example scientists may use such queries to query multiple scientific data sources, or a law enforcement agency may use it in order to find rare associations from multiple data sources. If standard query semantics were applied to such queries, all but the most trivial queries will return an empty answer.

By contrast, MystiQ relies on a probabilistic query semantics, and returns, along with each answer, the probability of it being what the user wants. It ranks the answers by this probability, and returns them to the user. MystiQ starts by assigning probabilities to all data items in all sources it queries. These probabilities can either be *static* or *dynamic*. Static probabilities are query-independent and are precomputed by the system and stored in the relational database. Constraint violations and fuzzy matches between objects generate static probabilities. To reach even further, the system also uses non-traditional ways to answer queries, such as statistics on the data sources, soft constraints extracted from the data, or data mining results. This enables it to return even more tuples, or adjust the probabilities of the tuples it returns. Each of them results in static probabilities. Dynamic probabilities are query-dependent, and are computed based on how well tuples in the data match the approximate predicates in the query. Starting from probabilities associated to the data items, MystiQ uses probabilistic database query plans: these plans are mapped back into SQL, and can be executed by any relational database engine. The technique is based on the theory of query evaluation on probabilistic databases [4].

This demonstration will illustrate the following features of MystiQ:

- support for complex SQL queries with approximate match predicates, and their ranked results.
- ability to return best matches when no tuple satisfies all the predicates
- support for complex SQL queries over inconsistent data, and how the system lowers the score of answers that use contradictory data.
- A *global constraint* definition, which is not enforced but used for detecting and resolving inconsistencies.
- The definition of a *soft view* and how it is used in queries.

2. SYSTEM OVERVIEW

The system has four main components. A data modeling language (mDML), a data definition language (mDDL), a preprocessor and a query translation engine.

2.1 mDML

This is the MystiQ data modeling language. It consists of a fragment of the SQL with two additional constructs. The first is a new *approximate match* operator with the following syntax:

$$A \sim B$$

Here A is any attribute defined in the query and B is either an attribute or a value. The operator enables probabilistic match between A and B. For an exact semantics, we refer the reader to [4].

The second construct enables users to specify their confidence in the various query predicates. The syntax is:

$$A\ op\ B\ [\texttt{CONFIDENCE} = f]$$

Here, *op* is either a standard SQL comparison operator ($=$, $<$, $>$, *LIKE*) or the $\sim$ operator. This construct says that with a probability $1 - f$, the user will also be interested in tuples that do not satisfy the predicate.

Note that specifying approximate match is semantically different from specifying a low confidence value and they can even be used in conjunction.

2.2 mDDL

This is the MystiQ data definition language and has several new constructs over the SQL DDL.

Predicate Functions mDDL specifies the functions to be used for generating probabilities for approximate predicates. The syntax is as shown in the following examples:

```
REGISTER ON Movie.actor PREDICATE QGRAM-DIST
REGISTER ON Movie.genre PREDICATE SEM-DIST
```

This says that an approximate predicate on actor names should use a q-gram function. This will match, for instance, `‘Coppola’` and `‘Capolla’`. On the other hand, film genres should be matched using a function that gives semantic distance between words. This would match `‘Satire’` with `‘Comedy’`, for example. All of these functions are preloaded in the database engine.

Global Views The system allows the user to define global views, which represent important concepts present in many data sources. They are defined in the LAV (Local As View) data integration paradigm. In LAV, the global views are never populated. MystiQ, however, populates them with probabilistic facts.

For example:

```
CREATE GLOBAL VIEW GActor (
   actor_name  VARCHAR(50),
   dob         DATETIME
)
```

Each source may be related to one or more global views. For example the following statement tells the system that the table ActorIMDB is related to the global view GActor and shows the relationship.

```
CREATE RELATIONSHIP VIEW
    ActorIMDB(name, dateOfBirth) AS
        select actor_name, DOB
        from GActor

CREATE RELATIONSHIP VIEW
    ActorMD(actor_name, birth) AS
        select actor_name, DOB
        from GActor
```

The first relationship says that the name and dateOfBirth in ActorIMDB can be obtained from GActor. Similarly for the second relationship view.

Global constraints

Global constraints are what the system believes to be true but are not enforced. An example is a constraint that says that each actor has a unique DOB (date of birth). The system may come across two data sources that contain different values of DOB for the same actor. Since global constraints are not enforced, both data sources can coexist in the system. MystiQ incorporates the global constraints in its probabilistic model and treats the contradictory information as less authentic. The exact procedure for doing this is described in the next section. The syntax for global constraints is defined below. Global constraints are specified on global views.

```
GLOBAL CONSTRAINT ON GActor actor_name → DOB
```

Soft Constraint Also called *statistics*, these are constraints that are believed to be true only in an expected sense. Consider global view `GActor(actor_id, fname, lname, dob)` and `GMovie(movie_id, title, director, writer, genre, year)`. Then, we have the following soft constraints:

```
SOFT CONSTRAINT ON GActor
  fname, lname ⇝ actor_id [RATIO 1.1]

GLOBAL CONSTRAINT ON GFilms
  director, writer ⇝ genre [RATIO 1.4]
```

The first constraint says that on average there are less than 1.1 actors with a given name i.e., (fname,lname) is a *soft key* for the actors table. Similarly, the second constraint says that `director` and `writer` almost determine the `genre`. MystiQ exploits soft constraints, when they are available.

Soft constraints differ from global constraints because their violation does not necessarily mean errors in data. MystiQ does not extract soft functional dependencies; these need to be extracted using other systems [6], then input manually.

2.3 Preprocessor

The preprocessor generates additional relational tables in the database with static probabilities given mDDL specification. The static probabilities are query-independent and based on constraint violations, statistics, and outputs of record linkage algorithms. The details of the technique can be found in [3].

The preprocessor also creates a set of *soft views*. A soft view is a view whose tuples are probabilistic, rather than deterministic. It can be used by the user for querying, and by the query processor. Soft views are *materialized.* We illustrate them with the following example:

```
DEFINE SOFT VIEW SportsMovie AS
  SELECT *
  FROM MovieIMDB
  WHERE title ~ 'sport'    or
        category ~ 'sport' or
        tagline LIKE '%football%'
```

This results in a probabilistic table that the user can query directly. Soft views are also employed to use the output of external record linkage algorithms. Consider two databases `IMDB` and `MD`, both containing an `Actors` table. A record linkage algorithm can be run to match actors in the two tables, which outputs pairs of actors along with a matching score. This can be stored as a soft view and used in answering queries that require information from both the databases.

2.4 Query Translation Engine

This is the central component that translates the user queries in mDML into standard SQL queries. It is based on our techniques for efficient query evaluation on probabilistic databases [4]. The translated query contains all the probability calculations embedded in the query as standard aggregates and can be directly executed by any standard relational engine to return the result tuples along with their probabilities. A useful feature of the translation engine is that the new query directly refers the tables that are in the original query. It does not require creation of intermediate probabilistic tables and hence, is efficient.

3. THE DEMONSTRATION CONTENT

We will demonstrate MystiQ by jointly querying four independent data sources: Internet Movie Database (IMDB) [7], Movie Database[9], KnowItAll [5], and WordNet [11]. IMDB is a clean, rich dataset about movies, actors, producers, quotes, etc. The Movie Database is noisier and smaller but has some extra information. The KnowItAll data is the result of a web information extraction project, which has collected concepts from the Web and populated them with instances. These concepts include scientists, politicians, cities, hospitals, etc. WordNet is the English language ontology.

The demo will allow the users to formulate complex SQL queries over these databases, like the examples below:

Example 1 Find an actor named Kevin whose first *successful* movie appeared in the year 1995.

```
SELECT *
FROM ACTOR A
WHERE A.name ~ 'Kevin'
  and 1995 =
        SELECT MIN(F.year)
        FROM Film F, CASTS C
        WHERE C.filmid = F.filmid
              and C.actorid = A.actorid
              and F.rating ~ "high"
```

Example 2 Find a film with title like 'Rain man', directed by 'Copolla' and year around 1975.

```
SELECT F.title, D.name, F.year
FROM   Director D, Films F
WHERE  D.did = F.did
   and D.name  ~ 'Copolla' [CONFIDENCE 0.9]
   and F.title ~ 'rain man'
   and F.year  ~ 1975      [CONFIDENCE 0.7]
```

The user has a high confidence in the director name and a lower confidence in the year. This is reflected in the query formulation.

Example 3 'List the movies based on the life of some scientist, along with the corresponding scientist names'. Note that the Scientists table here is a probabilistic table, being a part of the KnowItAll data. This query will use the 'role' field of the Casts table, which contains the type and/or name of the character for each (actor,movie) pair, in an approximate join.

```
SELECT F.FILMNAME, S.NAME
FROM Films F, Casts C, Scientists S
WHERE C.ROLE ~ S.NAME
and  C.filmid = F.filmid
```

Example 4 (Constraint Violations) We illustrate this using a simple query: *return all films made after 1990 which cast an actor born before 1940.*

We use two different movie databases that have inconsistent dates of birth for some actors. We show that if for some film, the only actor that was born before 1940 has inconsistent information in the other database, the movie is returned but with a lower score.

4. RELATED WORK

Several systems have been proposed in the past that compute ranked answers to queries based on fuzzy matches [8, 1], similarity joins [1], ontology and semantic similarity [10], etc. However, these systems don't use probabilities, the ranking functions are ad hoc and the scores lack any properly defined meaning. Thus, these scores cannot be combined, and used in conjunction. Also, its extremely hard to re-engineer the ranking function if a new source of uncertainty arises. Current approaches [2] to answering queries in inconsistent databases do not assign scores to output tuples and further suffer from lack of practical evaluation algorithms.

5. CONCLUSIONS

MystiQ significantly enhances the querying capability of a typical relational DBMS by allowing a variety of approximations to be added to normal SQL queries. It provides a powerful means to query dirty, inconsistent data across multiple data sources. Our preliminary evaluation indicates it successfully returns relevant and properly ranked results to various kinds of queries. It is a working prototype for an exciting, new querying paradigm.

6. REFERENCES

[1] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *CIDR*, 2003.

[2] L. Bertossi, J. Chomicki, A. Cortes, and C. Gutierrez. Consistent answers from integrated data sources. In *International Conference on Flexible Query Answering Systems*, 2002.

[3] Nilesh Dalvi, Jihad Boulos, Shobhit Mathur, Chris Re, and Dan Suciu. A probabilistic approach to inconsistent data, University of Washington Technical Report 2005.

[4] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004.

[5] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW*, pages 100–110, 2004.

[6] Ihab F. Ilyas, Volker Markl, Peter Haas, Paul Brown, and Ashraf Aboulnaga. Cords: automatic discovery of correlations and soft functional dependencies. In *SIGMOD*, pages 647–658, 2004.

[7] Internet movie database: *http://www.imdb.com/.*

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

[9] Movie database: *http://kdd.ics.uci.edu/database-s/movies/movies.html.*

[10] Anja Theobald and Gerhard Weikum. The xxl search engine: ranked retrieval of xml data using indexes and ontologies. In *SIGMOD*, pages 615–615, 2002.

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