SMARTINT: Using Mined Attribute Dependencies to Integrate Fragmented Web Databases

Ravi Gummadi [1], Anupam Khulbe [2], Aravind Kalavagattu [3], Sanil Salvi [4], Subbarao Kambhampati [5*]
Department of Computer Science, Arizona State University, Tempe, AZ, USA
[1]gummadi@asu.edu [2]akhulbe@asu.edu [3]aravindk@asu.edu [4]sdsalvi@asu.edu [5]rao@asu.edu

ABSTRACT

Many web databases can be seen as providing partial and overlapping information about entities in the world. To answer queries effectively, we need to integrate the information about the individual entities that are fragmented over multiple sources. At first blush this is just the inverse of traditional database normalization problem - rather than go from a universal relation to normalized tables, we want to reconstruct the universal relation given the tables (sources). The standard way of reconstructing the entities will involve joining the tables. Unfortunately, because of the autonomous and decentralized way in which the sources are populated, they often do not have Primary Key - Foreign Key relations. While tables do share attributes, direct joins over these shared attributes can result in reconstruction of many spurious entities thus seriously compromising precision. We present a unified approach that supports intelligent retrieval over fragmented web databases by mining and using inter-table dependencies. Experiments with the prototype implementation, SMARTINT, show that its retrieval strikes a good balance between precision and recall.

See *arxiv.org/abs/1101.5334* for a longer version of this paper

Categories and Subject Descriptors

H.3.5 [INFORMATION STORAGE AND RETRIEVAL]: Online Information Services|Web-based services

General Terms

Algorithms, Design, Experimentation.

1. INTRODUCTION

An increasing fraction of the information accessible on the web comes from a welter of uncurated web databases, which provide partial but overlapping information about entities in the world. In database terms, web sources can be seen as exporting parts of a universal relation describing the entities of interest. There are however two important changes from the traditional database setup: (i) the database administrator, who ensures lossless normalization, is replaced by *independent data providers* and (ii) specialized users, who are aware of database querying language, are replaced by *lay users*. These changes in turn have two important implications:

Ad hoc Normalization by providers: Primary key-Foreign key (PK-FK) relationships that are crucial for reconstructing the universal relation are often missing from the tables. This is in part because partial information about the entities are independently entered by data providers into different tables, and synthetic keys (such as vehicle ids, model ids, employee ids) are simply ignored. In some cases, such as public data sources about people, the tables may even be explicitly forced to avoid keeping such key information.

Imprecise queries by lay users: Most users accessing these tables are lay users and are not often aware of all the attributes of the universal relation. Thus their queries may be "imprecise" [2] in that they may miss requesting some of the relevant attributes about the entities under consideration.

Thus a core part of the source integration on the web can be cast as the problem of reconstructing the universal relation in the absence of primary key-foreign key relations, and in the presence of lay users. In order to find relevant attributes which describe the entity, and provide complete information about that entity to users, we require: (i) Linking attributes and propagating constraints spanning across multiple tables, and retrieving precise results. (ii) Increasing the completeness of the individual results by retrieving additional relevant attributes and their associated values from other overlapping tables not specified in the query (thereby reconstructing the universal relation from different local schemas).

SMARTINT aims to provide effective solutions to these challenges. It starts with a base table containing a subset of query-relevant attributes, and attempts to "complete" the tuples by predicting the values of the remaining relevant attributes. Intuitively, the base table should contain important attributes whose values cannot be predicted accurately, but which can help in predicting the values of the other relevant attributes. The prediction/completion of the tuples is made possible by approximate functional dependencies (AFDs). To illustrate, on a vehicles database, AFDs are rules of the form $\{model\} \rightsquigarrow vehicle\_type$, $\{model\} \rightsquigarrow review$ etc. They allow us to expand partial information about the car model into more complete information about vehicle type and review, as well as propagate constraints in the query on vehicle type/review.

*This research is supported in part by the NSF grant IIS-0738317, the ONR grant N000140910032 and a Google research award.

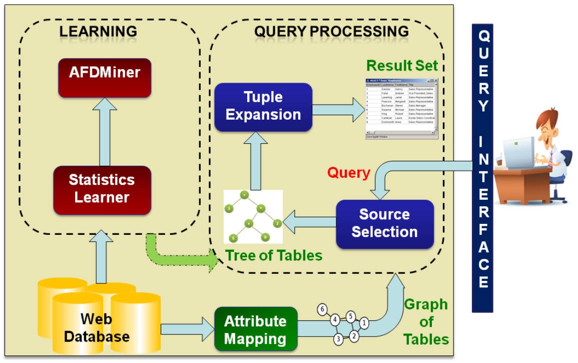
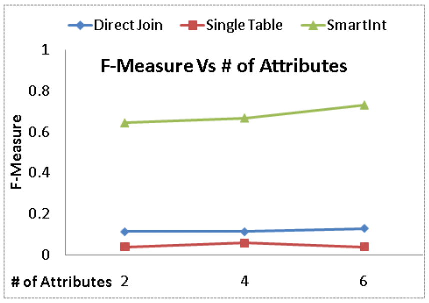


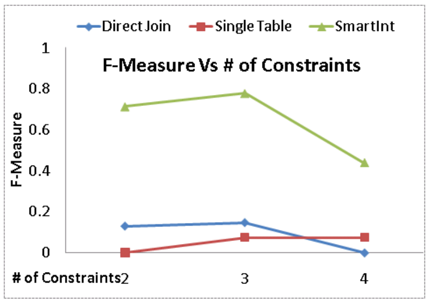
Figure 1: Architecture of SMARTINT System

2. SMARTINT ARCHITECTURE

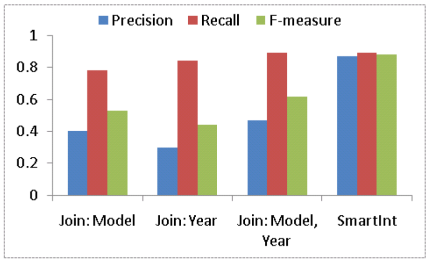
Figure 1 shows the architecture of SMARTINT. The system has two components: (i) a *learning component,* which mines AFDs and source statistics from different sources and (ii) a *query answering*



(a)



(b)



(c)

Figure 2: (a) F-measure *vs* Number of Attributes, (b) F-measure *vs* Number of Constraints and (c) SMARTINT *vs* Multiple join paths

component which actively uses the learned statistics to propagate constraints and retrieve attributes from other non-joinable tables.

The Query Answering Component

Given a query, this *online* component focuses on selecting the most appropriate tables and processing them to get the result set.

Source Selector: The first step, after the user submits the query, is to select most relevant tables from the potentially large number of tables present in the database. Source selector outputs a 'tree of relevant tables'. In order to construct the tree, it first picks the top n tables (where n is the maximum size of the tree). Then it evaluates the relevance of all the subgraphs of tables of size k and picks the most relevant subgraph. It then picks the most relevant tree within that graph and returns it. Estimating the relevance of tables with respect to the query during query processing can be costly. SMARTINT uses 'Source Statistics' learned beforehand to calculate the degree of relevance of a table for a particular query.

Tuple Expander: The next step in the query answering phase is to use the 'tree of tables' returned by the source selector to construct the final result set. Tuple expander first constructs the hierarchical schema from the table tree. It uses AFDs to determine which attributes from the child table are appended to the schema. Once the schema is constructed, it starts populating the tuples corresponding to this schema. It first queries the 'root table' from the 'table tree' and then starts appending the determined attributes using the stored values from 'Source Statistics' module.

The Learning Component

This *offline* component learns the AFDs and source statistics used by the query processing component.

AFD Mining: We define and mine AFDs as condensed representations of association rules. For example, an AFD (*Model⇝Make*) is a condensation of association rules (*Model:Accord⇝Make:Honda*), (*Model:Camry⇝Make:Toyota*) *etc.* Our search for high quality AFDs is guided by two metrics, namely *confidence* and *specificity*, which are analogous to the standard confidence and support metrics used in association rules. *AFDMiner* [1] performs a bottom-up search in the attribute lattice to find all AFDs and FDs that fall within the given *confidence* and *specificity* thresholds. We use the AFDs learnt within each table along with the attribute mappings (which serve as anchor points) to learn rules across tables. While combining AFDs across tables, we multiply their confidence to get the confidence of the final rules and pick the best.

Stat Learner (Source Statistics): Source statistics are extensively used in both 'Source Selector' and 'Tuple Expander'. It might seem that Stat Learner would require another scan of the database to mine useful statistics. But since we mined AFDs by rolling up association rules, and confidence of an association rule is nothing but the conditional probability of the attribute-value pairs involved, we store them during the AFD mining process. Apart from these, we also store information related to value distribution in each column to calculate the extent of overlap between two tables. This helps us in approximating the relevance measure during query time.

3. EVALUATION

A prototype implementation of SMARTINT has been completed and demonstrated at ICDE 2010 [3]. To evaluate SMARTINT, we used a vehicles database with 350,000 records probed from GOOGLE BASE. We created a *master table* with 18 attributes, and divided it into multiple child tables with overlapping attributes. We compared the accuracy of SMARTINT with '*Single table*' and '*Direct join*' approaches. In the single table approach, results are retrieved from a single table which has maximum number of attributes (constraints) mentioned in the query mapped on it. The direct join approach involves joining the tables based on the shared attributes. We measured the value of precision and recall by varying the number of projected attributes and number of constraints in the query.

The plots in Figure 2 show the results of our evaluation. Plots Figures 2 (a) and 2 (b) show that SMARTINT scores higher on F-measure compared to single table and direct join methods. In cases where query constraints span over multiple tables, single table approach ends up dropping all the constraints except the ones mapped on to the selected best table. This results in low precision. Despite dropping constraints, it still performs poorly on recall since the selected table does not cover all the query attributes, and hence answer tuples are low on completeness. Although Direct join approach, which allows combining partial results from selected tables over all shared attributes, provides high recall, it suffers from low precision. Specifically, in the absence of primary-foreign key relationships, it ends up generating non-existent tuples through replication. In contrast, SMARTINT processes the distributed query constraints effectively using the mined attribute dependencies and hence keeps the precision fairly high. At the same time, it performs chaining across tables to improve the recall. Figure 2 (c) shows that SMARTINT had higher F-measure than all possible join paths. The results demonstrate that SMARTINT achieves a better balance between precision and recall.

4. CONCLUSION

We presented a unified approach that supports intelligent retrieval over fragmented web databases by mining and using inter-table dependencies. Our experimental results demonstrate that approach used by SMARTINT is able to strike a better balance between precision and recall than can be achieved by relying on single table or employing direct joins. Additional details about our approach and evaluation can be found in *arxiv.org/abs/1101.5334*.

5. REFERENCES

[1] A. Kalavagattu. Mining approximate functional dependencies as condensed representations of association rules. Master's thesis. Arizona State University. 2008.

[2] U. Nambiar and S. Kambhampati, "Answering imprecise queries over autonomous web databases," in *ICDE*, 2006, p. 45.

[3] R. Gummadi, A. Khulbe, A. Kalavagattu, S. Salvi, and S. Kambhampati, "SmartInt: A system for answering queries over web databases using attribute dependencies," ICDE 2010 (Demo).