Semantic Search of Schema Repositories

T. Syeda-Mahmood
Gauri Shah
IBM Almaden Research Center
650 Harry Road
San Jose, CA

Lingling Yan
IBM SVL
555 Bailey Avenue
San Jose, CA 95141

Willi Urban
IBM Software Group
Schoenaicher Str. 220
Boeblingen 71032

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms

1. INTRODUCTION

With XML fast becoming the de facto standard for representing structured metadata in databases and internet applications, there is a rise in the need for efficient search mechanisms for the searching such repositories in several application domains. In this poster, we outline the requirements of a search engine, and lay the theoretical foundations for a fast and efficient search mechanism for these XML (and other) repositories.

We formulate the problem of finding matching schemas as the problem of computing a maximum matching in the pairwise bipartite graph formed from the query and repository schema attributes. The edges of the bipartite graph capture the similarity between corresponding attributes in the schema. To ensure meaningful matches, we use both name and type semantics in modeling attribute similarity. Since detailed graph matching is compute-intensive, our approach uses upper and lower bounds on the size of the matching to prune candidate schemas. Finally, we develop a technique for schema indexing called *attribute hashing* for fast database schema indexing. The matching schemas of the database are then found by indexing the hash table using query attributes, performing lower bound computations for maximum matching, and recording peaks in the resulting histogram of hits. The key rationale used is that related schemas in the database have an overwhelming number of attributes semantically-related to query attributes so that indexing based on query attributes could only point to relevant matching schemas.

2. SEARCH AS RANKED GRAPH MATCHING

In ranked graph matching, we want as many of the query attributes to match the repository schema attributes with very few unmatched candidates left on both sides. We desire a graph matching of maximum cardinality as well as maximum weight. Consider a bipartite graph $G = (V = X \cup Y, E, C)$ where $X \in Q$ and $Y \in D$ are attributes in the query and repository schemas Q and D respectively, E are the edges, and $C : E \rightarrow \mathbb{R}$ are the similarity scores representing similarity between query and schema attributes per edge. An edge is drawn between two attributes only if they are semantically related.

The ranking of a schema is then given as $R_1(D) = 2 \cdot |M_D|/(|Q|+|D|)$ where M_D is a maximum-cardinality matching in the schema D. For schemas that have the same rank R_1, they are further ranked by $R_2(D) = C_{max}(M_D)/|M_D|$ where $C_{max}(M_D)$ is the maximum-similarity score associated with the maximum matching M_D. In practice, we retain all matches that are above a chosen threshold T.

Although, algorithms are available for computing the maximum-cardinality, maximum-weight bipartite graph matching [1], finding a maximum matching of maximum weight is a compute-intensive operation. To speed it up, we derive tight upper and lower bounds on the size of the matching that can be quickly computed, and use the bounds for ranking purposes.

3. CAPTURING SEMANTIC SIMILARITY BETWEEN ATTRIBUTES

The above method is independent of the method used to determine the relationship between query and repository schema attributes. To ensure meaningful matches, we use both name and type semantics in modeling similarity between attributes. We capture name semantics using a technique similar to the one in [2] detailed as follows:

Word tokenization: To tokenize words, we find word boundaries in a multi-term word attribute using changes in font, presence of delimiters, etc.

Part-of-speech tagging and filtering: Simple grammar rules are used to detect noun phrases and adjectives. Stop-word filtering is performed using a pre-supplied list.

Abbreviation expansion: The abbreviation expansion uses multiple vocabularies. Multiple expansions for words and their synonyms are retained for processing.

Synonym search: We use the WordNet thesaurus [3] to find matching synonyms to words. Each synonym is assigned a similarity score based on the sense index, and the order of the synonym in the matches returned.

Match generation: Consider a pair of candidate matching attributes (A, B) from the query and repository schemas respectively. Let A, B have m and n valid tokens respectively. The semantic similarity between attributes A and B is then given by $Sem(A, B) = 2 \cdot \frac{Match(A,B)}{m+n}$ where $Match(A, B)$ are the matching word tokens between the two attributes.

WWW 2005, May 10–14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

4. ATTRIBUTE HASHING

Indexing of the repository schemas is crucial to reducing the complexity of search. We introduce and use a new method called *attribute hashing*, to index schemas that allows determination of valid edges of the bipartite graph to find the matching faster. We only give a high-level view of attribute hashing.

The basic idea of attribute hashing is to hash the feature list i.e. synonyms per word token of each attribute. All the attributes with the same feature list are hashed together in the same location. The hash table also maintains the indexing schema of these attributes such as word index within a schema, and schema index within the repository. Then given an attribute from the query schema, the matching attributes from repository schemas are obtained by computing the feature list of the query attribute, and indexing using the hash function.

5. SEMANTIC SEARCH ALGORITHM

We now describe our overall approach to searching schema repositories.

Schema Parsing: The first step in off-line index creation is to parse the metadata to create the schemas. We use different parsers based on the metadata types.

Offline index creation: The parsers used to extract the schemas are also used to extract word attributes along with their tag types. We then separate multiple terms in each word into tokens, perform part-of-speech tagging and word expansion, and derive synonyms per token using the WordNet thesaurus.

Query processing: Query schemas are processed in a similar fashion to repository schemas except that no synonyms are looked up for the tokens of query attributes. The tokens are directly used to find matches to get closer matches than those obtained by looking up synonyms of synonyms.

Search algorithm: The search algorithm extracts the word tokens for each attribute of the query schema, and computes the semantic hash for each such token. It checks that the type tags of the hashed entries match, and updates the hit counts of the words from the schema repository. A semantic match of a query word to a repository schema word is indicated if enough number of tokens of the query word find a match to the repository schema word. When the words are found to be semantically related, the histogram of the schema hits is updated only if the degree counts of the corresponding attributes are 0. This ensures that each query word is accounted for only once in the matching repository schema. The resulting histogram is normalized to derive the schema rank.

6. RESULTS

The algorithm for searching for XML schemas was tested on a business object repository consisting of 517 business objects drawn from Crossworlds business object library designed for Oracle, PeopleSoft and SAP applications.

Figure 1 shows the average precision versus recall using three different methods of schema matching: full-text indexing, lexical matching, and our semantic matching for 20 query schemas. For each schema, we manually selected the ideal matching schemas from the whole database. We then ran the semantic match algorithm and counted the number of matching schemas for each threshold value. For comparison with other methods, we allowed as many schema matches as with the semantic match, and then computed the average precision and recall. We can see that the semantic match method performs much better than that other methods in the precision versus recall graphs.

We tested the indexing performance of the hashing scheme by noting the fraction of the database touched during search. Our experiments show that on average a 90-95% reduction in search is achieved by the indexing step.

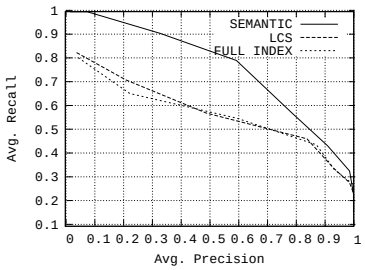

Figure 1: Average precision versus recall.

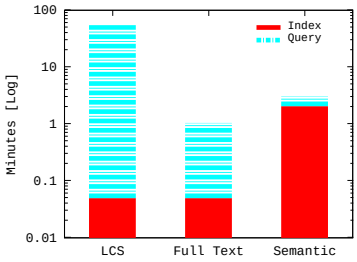

Figure 2: Time to index database and query it.

Figure 2 shows the time taken to run queries using the three different methods. We note that indexing the database using semantic matching takes a long time but this is a one-time offline requirement. Queries using semantic matching are much faster than the queries using full-text indexing and lexical matching.

7. CONCLUSIONS

In this poster, we have laid the theoretical foundations of searching through XML schema repositories for semantically-related schemas. We have captured semantic relationships coupled with fast indexing mechanisms. Comparison with full-text search has shown the semantic search method outperforms full-text search in both precision and recall while keeping the search time comparable.

9. REFERENCES

[1] Andrew V. Goldberg and R. Kennedy. An Efficient Cost Scaling Algorithm for the Assignment Problem. *SIAM Journal on Discrete Mathematics*, 6(3):443–459, Apr. 1993.

[2] J. Madhavan, P. Bernstein, K. Chen, A. Halevy, and P. Shenoy. Corpus-based Schema Matching. In *Proceedings of the Information Integration on the Web*, pages 59–66, Acapulco, Mexico, Aug. 2003.

[3] G. A. Miller. WordNet: A Lexical Database for the English Language. `http://www.cogsci.princeton.edu/ wn/`.