Searching XML Databases for Semantically-related Schemas

Gauri Shah
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120

Tanveer Syeda-Mahmood
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120

ABSTRACT

In this paper, we address the problem of searching schema databases for semantically-related schemas. We first give a method of finding semantic similarity between pair-wise schemas based on tokenization, part-of-speech tagging, word expansion, and ontology matching. We then address the problem of indexing the schema database through a semantic hash table. Matching schemas in the database are found by hashing the query attributes and recording peaks in the histogram of schema hits. Results indicated a 90% improvement in search performance while maintaining high precision and recall.

1. INTRODUCTION

A number of applications including business data warehousing and business process integration, require data consolidation and integration by system integrators and data warehouse specialists. Currently, this is done in a time-consuming fashion by manually looking at schemas and recording semantic relationships on spreadsheets. The problem of automatically finding semantic relationships between schemas has been addressed by a number of database researchers lately [2-10], where the focus has been on semantic schema matching of schema pairs. But when the number of schemas is large, it is impractical to use approaches such as similarity flooding [2], and other detailed matching approaches to find related database schemas in response to query. In this paper, we present an information retrieval technique for finding semantically related schemas in the database. Our approach is based on the rationale 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 point to relevant matching schemas. Finding semantic relationship between query attributes is difficult, in general, because (1) query attributes could be multi-word terms (e.g. Customer2Identification, PhoneCountry, etc. which require tokenization. Any tokenization must capture naming conventions used by DBAs, system integrators, programmers to form attribute names. (2) Finding meaningful matches would need to account for senses of the word as well as their part-of-speech through a thesaurus. (3) Finally, multiple matches of attributes must be taken into account.

2. Selecting matching schemas

Given a set of schemas in the database, it is reasonable to assume that the best matching schemas are those that have a large number of semantically related attributes. Assuming semantic relationship between attributes can be defined, let us consider how we may select the best matching schemas. Let there be k schemas in the database. Let $h_1, h_2, \ldots, h_k$ be the number of attributes of schemas $S_1, S_2, \ldots, S_k$ that have found a match to one or more of the R query attributes. Let $q_1, q_2, \ldots, q_k$ be number of query attributes that found a match to at least one attribute in schemas $S_1, S_2 \ldots, S_k$. Let $n_1, n_2, \ldots, n_k$ be the number of attributes present in schemas $S_1, S_2, \ldots, S_k$. Then the overall match of the query schema to a database schema is given by

$M_i = \mathit{min}\{h_i/n_i, q_i/R\}$..(1)

The best matching schema to a query schema is given by

$S_{best} = \mathit{max}\{M_i\}$ for all $1 \leq i \leq k$..……………………………(2)

Further, the values of M_i can be sorted to get a ranked list of matching schemas. By taking $\mathit{max}\{M_i\}$, we look for those matches that have the lowest number of unmatched attributes relative to their schema size.

2.1 Finding semantically related attributes

For the above operation to be meaningful, the individual attributes must be matched semantically. In this paper, we focus on capturing semantics through similarity in names of attributes taking into account their multi-word nature. Using a technique similar to the one in [3], we parse the words to find ontological similarity in their tokens. The parsing uses tokenization, part-of-speech tagging, filtering and abbreviation expansion to generate list of candidate words. Thus CustomerPurchase will be separated into Customer and Purchase. The tokenization uses font changes, underscores, spaces, numbers and other delimiters. Abbreviations such as CustPurch will be expanded into CustomerPurchase, CustomaryPurchase, etc, using a domain-dependent abbreviation expansion dictionary generated a priori. Filtering removes stop words and part-of-speech tagging classifies words as nouns, adjectives, etc. The resulting words are then used to index an ontology (Wordnet Thesaurus [1]) to obtain a list of synonyms. Consider a pair of candidate matching attributes (A, B). Let A, B have m and n valid tokens respectively, and let S_{yi} and S_{yj} be their expanded lists based on ontological processing. We consider each token i in source attribute A to match a token j in destination attribute B if $i \in S_{yi}$ or $j \in S_{yi}$. The semantic similarity between attributes A and B is then given by

$\mathrm{Sem}(A, B) = 2*\mathrm{Match}(A,B)/(m + n)$…………………………(3)

where Match(A, B) are the matching tokens based on the definition above. The semantic similarity allows us to match attributes such as (state, province), (CustomerIdentification, ClientID), (CustomerClass, ClientCategory), etc.

3. Indexing Schemas

A straightforward implementation of the above algorithms would not scale very well for large schema databases. For example, in a database of 500 schemas, a schema could have over 50 attributes, and 2-5 tokens per attribute, and 5-30 synonyms per token, making the search for a query of 50 attributes easily around 50 million operations per query. To enable indexing, we developed a semantic hash table for the schema database. Specifically, the synonyms for tokens derived from all attributes in all schemas are used as keys of a semantic hash table that records 3-tuple indices $\{(t_i,w_j,s_k)\ldots\}$ indicating the index of the token, the attribute from which the token is derived, and the schema from which the attribute is derived.

Given a query, the best matching schemas are found as follows. Each token of the query attribute is used as the key to index the semantic hash table. All the hit counts of attributes present in the hash table entries are updated by 1. When all tokens of the query are processed, the attribute match score is computed using Equation 3. If the semantic relationship score is above a threshold T, then the match is retained and the schema hit is updated by 1. The attribute hit of the query for that schema is updated by 1 as well. We chose T=0.67 based on results of user studies in which a large fraction of people chose those attributes to be semantically related if at least 2/3rd of their tokens were related. After all the query attributes are processed, the final scores of matching schemas are computed using Equation 1 and 2. The result is a ranked list of matching schemas. By using a suitable threshold T2, different tradeoff between precision and recall could be demonstrated.

4. Results

We tested the performance of semantic schema retrieval on a business object database consisting of 517 application-specific and generic business objects drawn from Crossworlds business object library. The business objects tend to have a larger number of member attributes (over 100). Further, there is frequently schema embedding in the XSD documents describing the schemas. Table 1 shows a subset of the matching attributes in a query and its matching database schema. As can be seen, semantic match of attributes allows for term matches when words are out of order, abbreviated, or have close meanings.

SCHEMA: PaymentInformation	SCHEMA: Email_PurchaseOrder
PaymentMethod	Hdr.PaymentTerm
PaymentAmount	Summary.TotalAmount
CreditCardType	Hdr.Payment.CInfo.CardType
CreditCardNumber	Hdr.Payment.CInfo.CardNum
NameOnCredictCard	Hdr.Payment.CInfo.CardHolderName
CreditCardApproval	Hdr.Payment.CInfo.CardAuthCode

Table 1: A subset of the attributes of two sample matching schemas.

We tested the indexing performance of the hashing scheme by noting the fraction of the database touched during search. Using the semantic hash table, the complexity of search reduces significantly, as only matching tokens are explored. In fact, our experiments show that on average a 90- 95% reduction in search is achieved by the indexing step. The entire 517 schema database consisting of over 100,000 total attributes indexes in less than 2 minutes on an Intel M-Pro 2 GHz Pentium, and the matching schemas for queries are retrieved instantaneously.

Table 2 shows the performance for sample query schemas. As can be seen, the matching schemas are in close agreement in the number of matching attributes. It can also be notes that only 3-5% of the database tokens are touched in the semantic hash table.
Next, we varied the threshold T2 to see the effect on precision and recall. The precision-recall curves for sample queries are illustrated in Figure 1. The average precision-recall curve over all queries tested is also shown in this figure (as thick line). From this figure, we can note that with a suitable choice of threshold it is possible to get an average precision of 85% with a recall of over 90%.

5. Conclusions

In this paper, we have presented an approach to search for semantically related schemas in the database, in response to queries. The indexing of the database using a semantic hash table generated from the synonyms of tokens of schema attributes allows for increased time performance without sacrificing precision and recall.

6. REFERENCES

[1] G.A. Miller. Wordnet: A lexical database for English. Communications of the ACM, 38(11):39-41, 1995.

[2] S. Melnik, H. Garcia-Molina and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In Proc. ICDE 2002.

[3] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In The VLDB Journal, pages 49-58, 2001.

[4] S. Bergamaschi, S. Castano, M. Vincini and D. Beneventano. Semantic Integration of Heterogeneous Information Sources. In Data and Knowledge Engineering, 36(3):215-249, 2001.

[5] W. Li and C. Clifton. SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases using Neural Networks. In Data and Knowledge Engineering, 33(1):49-84, 2000.

[6] A. Doan, P. Domingos and A. Halevy. Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In Proc. SIGMOD 2001.

[7] H. Do and E. Rahm. COMA: A System for Flexible Combination of Schema Matching Approaches. In Proc. VLDB 2002.

[8] A. Doan, J. Madhavan, P. Dominogos and A. Halevv. Learning to Map between Ontologies on the Semantic Web. In Proc. WWW 2002.

[10] E. Rahm and P. Bernstein. A Survey of Approaches to Automatic Schema Matching. In VLDB Journal, 10(4):334-350, 2001.

[11]. B. He and K. Chang. Statistical Schema Matching across Web Query Interfaces. In Proc. SIGMOD 2003

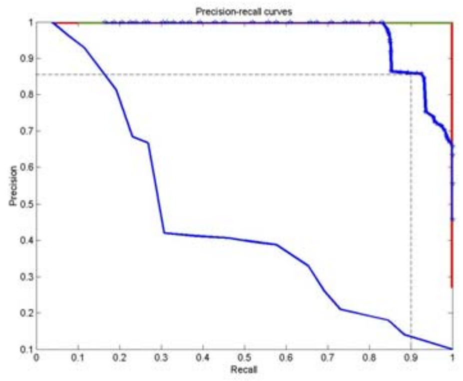
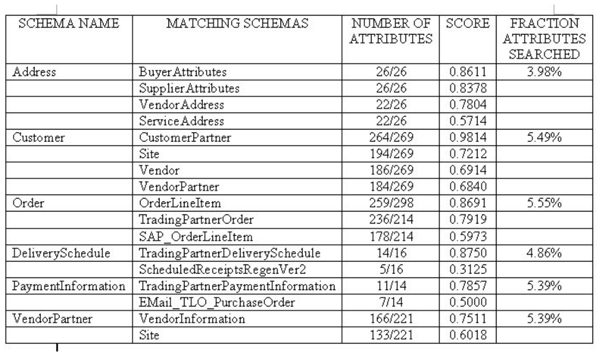


Figure 1: Precision-recall curves for queries. The dark curve is the average precision and recall across all queries tried...



SCHEMA NAME	MATCHING SCHEMAS	NUMBER OF ATTRIBUTES	SCORE	FRACTION ATTRIBUTES SEARCHED
Address	BuyerAttributes	26/26	0.8611	3.98%
	SupplierAttributes	26/26	0.8378	
	VendorAddress	22/26	0.7804	
	ServiceAddress	22/26	0.5714	
Customer	CustomerPartner	264/269	0.9814	5.49%
	Site	194/269	0.7212	
	Vendor	186/269	0.6914	
	VendorPartner	184/269	0.6840	
Order	OrderLineItem	259/298	0.8691	5.55%
	TradingPartnerOrder	236/214	0.7919	
	SAP_OrderLineItem	178/214	0.5973	
DeliverySchedule	TradingPartnerDeliverySchedule	14/16	0.8750	4.86%
	ScheduledReceiptsRegenVer2	5/16	0.3125	
PaymentInformation	TradingPartnerPaymentInformation	11/14	0.7857	5.39%
	EMail_TLO_PurchaseOrder	7/14	0.5000	
VendorPartner	VendorInformation	166/221	0.7511	5.39%
	Site	133/221	0.6018	

Table 2: Sample query schemas with matches from database schemas.