2008 International Conference on Computer Science and Software Engineering

Translating Query for Deep Web Using Ontology*

Hao liang
1. College of Computer Science and Technology, Jilin University
2. Department of Information Science and Technology, Changchun Taxation College, Changchun 130117, China
Email: liangh434@163.com

Wanli Zuo
1. College of Computer Science and Technology, Jilin University
2. Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun 130012, China
Email: wanli@jlu.edu.cn

Fei Ren
1. College of Computer Science and Technology, Jilin University, Changchun 130012, China
Email:Renf854@163.com

Junhua Wang
1. College of Computer Science and Technology, Jilin University
2. Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun 130012, China

Abstract

The E-commerce information on the Surface Web is supported by the Deep Web, which can not be accessed directly by the search engines or the web crawlers. The only way to access the backend database is through query interface. Extracting valid attributes from the query forms and automatic translating the source query into a target query is a solvable way for addressing the current limitations in accessing Deep Web data sources. To generate a valid query, we have to reconcile the key attributes and their semantic relation. We present our framework to solve the problem. To enrich the set of attributes contained in the semantic form, we use the WordNet as kinds of ontology technique and we try to find the semantic relation of the attributes in the same query from and different forms. Extensive experiments over real-word domains show the utility of our query translation framework.

Keywords: Surface Web; Deep Web; WordNet; ontology.

I. INTRODUCTION

With the explosive growth of the Internet, an increasing number of databases are becoming accessible through search interfaces, and many of these sources are E-commerce sites supported by database. These databases are called Deep Web, which can not be crawled by the search engines. The Web has been rapidly "deepened" by massive hidden databases. While the Surface Web has linked billions of static HTML pages, a far more significant amount of information is believed to be "hidden" in the Deep Web, behind the query forms of searchable databases. A survey in April 2004 estimated there were more than 450,000 online databases [1]. Myriad information may not be accessed through static URLs because they are presented as result after users submitted the query. The Deep Web databases require manual query interfaces and dynamic programs to access their contents, thus preventing Web crawlers from automatically extracting their contents and indexing them, and therefore not being included in search engine results [2]. For integrating the resources of Deep Web, we have to find the accessible query interfaces and integrate them.

While there are so many databases online, users have difficulties in finding the right ones and get information from databases. We need find online databases and then integrate the databases provide uniform query interface for getting practical information. Integrating the Deep Web faces some new challenges. First, the Deep Web is a large number of queryable and accessible distributed discarded widely. Second, such integration is dynamic because the sources maybe proliferated and evolved on the web, they cannot be statically configured. Third, it is ad-hoc. Queries maybe submitted by users for different aims [3]. Due to the semi-structured nature of HTML data and the heterogeneities of the sources, significant laborious human efforts are involved in the building process, especially when the number of sources is large. As a result, building such a system is time-consuming and needs lots of expertise. We try to generate query across different resources of which the query interfaces are semantically the same.

We need develop techniques to get query across different Deep Web sources which mean that we have to translate queries without primary knowledge. Some methods can be concerned such as type-based search-driven translation framework by leveraging the "regularities" across the implicit data types of query constraints. In [3] they found that query constraints of different concepts often share similar patterns, and encoded more generic translation knowledge for each data type. They provided an extensible search-driven mechanism. We propose an attribute search-driven mechanism, in this demo the most important factor is the attributes and semantic relations between them. We try to extract abundant attributes, which describe the concept, and the relationships between the set of attributes of same search form and even different forms. The most efficient and effective technique of detecting the semantic relation between words is the WordNet [4]. We extend each attribute into a concept set which is used for matching attributes.

II. THE QUERY TRANSLATING FRAMEWORK

The query translation framework is composed of two steps: valid attribute extraction and query translating. The framework takes source query forms as valid input resources and generates a query form as output for target query. During the translation,

* The Science and Technology Development Program of Jilin Province of China under Grant No.20070533

978-0-7695-3336-0/08 $25.00 © 2008 IEEE
DOI 10.1109/CSSE.2008.630

we first extract valid attributes from query forms and find the semantic relation between attributes, and then compose attributes according to the web semantic restriction, finally rewrite the query for target form. Then users can query against the global query interface to get information exactly.

2.1 Attribute extraction

The inner identifiers can be easily obtained from HTML elements by a program, but they can not be directly used for further analysis. We need to do some additional process works, because the inner identifiers are usually comprised of several words and symbols. The IIS, which is shorted for inner identifier set, should be condensed into more general words. The Algorithm1 shows steps for separating a set of inner identifiers of a web data source.

Obtaining the valid attributes involves additional difficulties. (1) Some candidate attributes in candidate set are abbreviations. For example, "dep" and "yr" are widely used to indicate "departure" and "year." (2) Some candidate attributes in candidate set appear in different forms (singular and plural) for example, "adults" and "adult." So some pre-processing tasks should go ahead. So in the algorithm 1 there is a pre-process function step.

Algorithm 1: Keyword set extraction
Step 1: Get IIS (inner identifier set) from web data source.
Setp 2: Remove duplicated in the IIS.
Step 3: Remove special symbols from IIS, generate more substrings.
special symbols (:, -, _, @, $, &, #, ?, !, *,etc.)
Step 4: Pre-process function, PPF for short.
Step 5: Extend the key words of IIS into a set by utilizing WordNet.
Step 6: Generate hierarchy tree for IIS.
Step 7: Refine the hierarchy tree .

Figure 1. Agorithm for Keyword set extraction

To extract valid attributes, there are some pre-processing tasks to be done, issues such as concatenated words, abbreviations, and acronyms are deal with [5, 6, 7, 8]. There are three steps to finish these pre-processing tasks:

Step 1: There are some information retrieval pre-processing method should be used, such as stopword removal and stemming. Some words we get from query form are no value, so removing stopword and stemming can ensure the valid matching of attributes.

Step 2: Expanding some abbreviations and acronyms to their full words, e.g., from "dept" to "department". The expansion procedure is done based on some domain information collected from other source form in the same subject.

Step 3: We break a label item into atomic words. Such items are usually concatenated words showed in the web pages.

For example, we can break "fromCity" into "from City". To make sure the validity， we need an English dictionary.

Finishing the pre-processing tasks, we get keywords, which are related to the attributes of the query form. Only some of them will be labeled as attributes. We take the "Book" domain as an example, the figure3 and figure4 show normal query forms in the domain, we take them as source query form and target query form.

In our demo, the 'att' is short for attribute, in the source query form there are four 'atts', and they are 'Author', 'Title', 'Age' and 'Price'. Take the attribute 'Author' as an example, we get its lemma and synonyms from WordNet and convert all the words into a set s_{Author} = {author| writer, maker, creator}. The marked value of s_{Author} is 'author', the others are the alias. The attribute set of a form is consisted of all the s_{Author}. The set of source form S= {s_{Author}, s_{Title}, s_{Age}, s_{Price}}.There are two reasons why choosing the 'author' as the marked value. It shows more frequently and has general meaning in the domain, and the word provides orientation information when we transfer the queries back. When getting synonyms from WordNet, we only consider the nominal meaning of attribute. We find the labels of query forms are absolutely nouns; the reason maybe is that the noun is enough to depict the information.

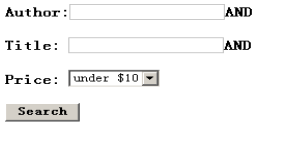

Figure 1. Source query form

Figure 3 Target query form

To translate queries, we need map the attributes of the target query form onto the sets. When mapping att_i of target query form onto S, we compare att_i with s_j if att_i exists in the values of s_j then we sent the value of s_j to att_i. There are three attributes should be set values according to S in the target form. They are 'writer', 'title' and 'price'. In general situation , we can automatically fulfill the target query form according to the source query form, and form a query containing 'author', 'title' and 'price' .

2.2 Finding the semantic constraints of the interface

There are two types of semantic constraints on the interface of Deep. First, some predicate templates may only be queried exclusively. For instance, target form allows only an exclusive selection among attributes 'writer', 'title' and 'ISBN'. Second, a form may have "binding" constraints, which require certain predicate templates be filled as mandatory. For instance, target form may require price not be queried alone and thus each form query must bind a predicate template from attributes. To solve this kind of problems, we need to find relation between attributes in the source and target query form with the help of semantic meanings of the web [9].

In the figure 2, the function of the form is to provide queries which contain only one of 'writer', 'title', 'subject' and 'ISBN'. The four query terms form a set, each time we can only query about one of them. It is necessary to find the relation between them when transferring the query for the target. It is obviously that we can find some evidences from the code of the target form. If the code is html style, then the figure 3 gives main description of the target form. The code shows that the 'writer', 'title', 'subject' and 'ISBN' are the same kind of the control panel and they share the container named 'RB1'.

We define a predicate container C= {att_1, att_2, att_3 …att_n:: relation} to present the attributes being constrained by the web semantics defined by the author of the web form. In C the ai presents the attributes be contained in the same container, relation presents the relation between the attributes. The C_{target}= {a_4, a_5, a_6:: exclusive } and C_{source} ={ a_1, a_2, a_3:: binding} ,which presenting in the table 1, show the exclusive and binding relation between the attributes.

```
<input id="RBIsbn" type="radio" name="RB1"
value="RBIsbn"/><label for="RBIsbn">ISBN</label>
<input id="RBtitle" type="radio" name="RB1"
value="RBtitle"/><label for="RBtitle">Title</label>
<input id="RBwriter" type="radio" name="RB1"
value="RBwriter"/><label for="RBwriter">Writer</
label><input id="RBSubject" type="radio" name="RB1"
```

Figure 4: HTML code of target query from

III. QUERY TRANSLATING

After extracting and mapping attributes, we get valid attributes for the query translation. This step is to generate valid query predicates from valid attributes. The query predicate is in a kind of template as <att, constraint, value>. Taking the attribute 'author' as an example, the predicate template is <author, like, 'Joanne' >. It is obviously that the attributes may have different data types like text, numeric and datetime. The predicate template of 'price' is <price, <, 35>, in this template we use '<', '>', '<=' and '>=' as the constraint.

In the source query form, user can use four attributes to describe a book, which means that the more attributes we have the more restrictive query predicate we can get.

When it comes to the target query form, user can use one of all the attributes to describe one facet of the book each time. To get translation of the different query forms, we have to get more valid predicates as we can. We can get some different valid predicate form the two query forms, which are P_1=<author, like, 'Joanne Kathleen Rowling' >and <price, <>, 0>, P_2=<title, like, 'Harry Porter and death saint machine'> and <price, <>, 0>.

When submitting P_1 or P_2 to the two query forms, the query results from two different Deep Web databases seem quite close. If we have some domain knowledge about book, we will find the 'price' is the least important attribute when describing a type of book. In the other domain, there are the same situations. When translating queries, it is better to make numeric attributes useless, because we have found the numeric attributes are not more important than the other text attributes.

IV. EXPERIMENT AND RESULT

The datasets used in our experiment come from MetaQuerier [3] DeepWeb dataset. This dataset contains the original query interfaces and their manually extracted query capabilities of 447 deep Web sources from 8 representative domains, which form 3 groups "TEL" in the Travel group: Airfares (49), Hotels (39), and Car Rentals (25); in the Entertainment group: Books(67), Movies (78), and Music Records (70); in the Living group: Jobs (52)and Automobiles (97). For each source, this dataset archives its root homepages and query-interface pages. In addition, it includes the manually extracted query capability for each interface.

We take a query form as the source query form and each of the same domain query forms as the target query form. Carry out our method between the two query forms. The result is showed in Table2. During the experiment we find that it is no necessary to fill all the form controls. Because of autonomy and heterogeneity of web databases, it is too hard to deal with some complex form controls and semantic relation between the forms. Especially in the Auto domain, there are so many attributes in the query forms, describing the attributes of a vehicle, and they are sometimes shows in very different types. The semantic relations between the attributes are very hard to handle. The situation leads to our low experiment results. There is still something we can do to improve the precise of translation.

TABLE I. RELATION OF ATTRIBUTES

	Source query form	**Target query form**
Candidate attributes	a_1=Author a_2=Title a_3=Price	a_4=Writer a_5=Title a_6=ISBN
Attributes mapping	a_1,a_2,a_3	a_1::a_4,a_2::a_5
Attributes semantic relations	Csource ={ a_1, a_2, a_3,:: binding}	Ctarget= {a_4, a_5, a_6,:: exclusive }

TABLE II. EXPERIMENT RESULTS

Domain	**Deep Web query forms**	**Correct Translation forms**	**Precise**
Airfares	30	26	86.7%
Books	30	26	86.7%
Automobiles	30	24	80%

V. DISCUSS AND FUTURE WORK

Our frame work illustrates an automatic process to extract the candidate attributes and translate queries between two interface query forms. The candidate attributes are extended into a set, which help to enhance the accuracy of mapping attributes. We propose to get more valid attributes by using the ontology technology, in this paper using WordNet, because we want to translate the query from one Deep Web to another quickly and without priori domain knowledge. The WordNet is used as a dictionary and semantic mapping mechanism.

It starts with a source and a target query interface specified by the user, and asks user to fill in a query in the source interface. The system then automatically translates the source query and fills in the target interface. We provide a method to solve the attribute heterogeneity during the translating queries. The traditional schema matching [10, 11, 12, 13] focuses on

mediating the heterogeneity at attribute level. Those works provide some concrete methods to form query translation assistant. Some approaches, e.g., [11, 12] require a collection of sources to mine the matchings, which are suitable for applications such as MetaQuerier. Others, e.g., [14, 15], perform matching across pairwise sources, which are suitable for applications such as a domain portal.

Ontology is a formal specification of a shared conceptualization [13]. The future work of our plan is to build domain ontology for some typical domain to collect domain knowledge and take domain ontology as priori knowledge to translate domain query. Domain ontology has been widely used in different field .There are a lot of work have been done about building ontology [12, 13, 16].

In this paper, we just use WordNet as taxonomy to get abundant attributes that is the foundational function of ontology. We want to get hierarchy relation between the same domain attributes and instance of them, by using the conceptual relation and instance relation we can describing the semantic relation of web page. Then users can query against the global query interface, which describing the semantic relation between attributes of the query interface.

VI. REFERENCES

[1] K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang, Structured databases on the web: Observations and Implications, SIGMOD Record, September 2004. 33(3):61–70

[2] UC Berkeley, Invisible or Deep Web: What it is, Why it exists, How to find it, and its inherent ambiguity, Available at http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/InvisibleWeb.html, July 2006.

[3] Bin He, Zhen Zhang, and Kevin C.-C. Chang, MetaQuerier: Querying Structured Web Sources On-the-fly, In Proceedings of SIGMOD 2005, System Demonstration, Baltimore, Maryland, June 2005.

[4] G. A. Miller. WordNet: A Lexical Database for English. Communications of the ACM, 38, 11 (Nov. 1995), 39-41.

[5] J., Madhavan, P.A.Bernstein, and E. Rahm, Generic Schema Matching with Cupid, In Proc 27th Int. Conf. on Very Large Data Bases (VLDB'01), 2001, pp. 49-58.

[6] Wensheng Wu, AnHai Doan, Yu, C., WebIQ: Learning from the Web to Match Deep-Web Query Interfaces, Proceedings of the 22nd International Conference on Data Engineering, ICDE '06, 2006.

[7] B. He and K. C.-C. Chang, Statistical schema matching across web query interfaces, SIGMOD Conference, 2003.

[8] A. Doan, P. Domingos, and A. Y. Halevy, Reconciling schemas of disparate data sources: A machine-learning approach, SIGMOD Conference, 2001.

[9] Berners-Lee, T., Hendler, J. and Lassila, O., The Semantic Web. Scientific American, 284 (5): 34-43, 2001.

[10] B. He, K. C.-C., Chang and J. Han., Discovering complex matchings across web query interfaces: A correlation mining approach, In SIGKDD Conference, 2004.

[11] Gruber TR, A translation approach to portable ontology specifications. Technical Report, KSL 92-71, Knowledge System Laboratory, 1993.

[12] DU Xiao-Yong, LI Man, WANG Shan, A Survey on Ontology Learning Research. Journal of Software, Vol.17, No.9, September 2006, pp.1837−1847

[13] Mathias Niepert, Cameron Buckner, Colin Allen, A Dynamic Ontology for a Dynamic Reference Work, JCDL'07, Vancouver, British Columbia, Canada, June 18–23, 2007.

[14] J. Kang and J. F. Naughton, On schema matching with opaque column names and data values, SIGMOD Conference 2003.

[15] A.Raja Raman, Y. Sagiv, J. D. Ullman, Answering queries using templates with binding patterns, In PODS Conference 1995.

[16] Marta Sabou, Chris Wroe, Carole Goble, Gilad Mishne, Learning Domain Ontologies for Web Service Descriptions: an Experiment in Bioinformatics. WWW 2005, May 1014, 2005, Chiba, Japan. ACM 1595930469/05/0005.