Ontology Based Automatic Attributes Extracting and Queries Translating for Deep Web

Hao Liang[1, 2]
1. College of Computer Science and Technology, Jilin University, Changchun 130012, China;
2. Department of Information, Changchun Taxation College, Changchun 130117, China
Email: liangh434@163.com

Fei Ren[3], WanLi Zuo[1, 4+], FengLing He[1, 4]
3. China Development Bank, Center of Operations, Beijing, 100037, China
4. Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun, 130012, China
Email: renf854@163.com wanli@jlu.edu.cn fengling@jlu.edu.cn

***Abstract*—Search engines and web crawlers can not access the Deep Web directly. The workable way to access the hidden database is through query interfaces. Automatic extracting attributes from query interfaces and translating queries is a solvable way for addressing the current limitations in accessing Deep Web. However, the query interface provides semantic constraints, some attributes are co-occurred and the others are exclusive sometimes. To generate a valid query, we have to reconcile the key attributes and semantic relation between them. We design a framework to automatically extract attributes from query interfaces taking full advantage of instances information and enrich the attribute sets embedded in the semantic query interface by Ontology technique. Each attribute is extended into a candidate attribute expressed by a hierarchy tree and describes the semantic relation of the attributes. We carry out our experiments in the real-world domain and results showed the validation of query translation framework.**

***Index Terms*—Deep Web, Surface Web, query interface, WordNet, Ontology, hierarchy tree**

I. INTRODUCTION

Recently, we witnessed the rapid growth of databases on the Web, which enriched the information of WWW. The World Wide Web should be divided into the Surface Web and the Deep Web [1]. The Surface Web consists of billions of searchable pages, while the Deep Web is hidden behind the Surface Web remaining unsearchable. A survey in April 2004 estimated there were more than 450,000 online databases [2]. 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.

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. He B, et al. found that query constraints of different concepts often share similar patterns, and encoded more generic translation knowledge for each data type [3, 4, 5]. They provided an extensible search-driven mechanism. We propose an attribute search-driven mechanism, in our framework the most important factor is the attribute and semantic relations between them. The schemas of Deep Web are composed of attributes in the query interfaces, so the validation and effectiveness of attribute extraction is the most important factor during the accessing to Deep Web. We try to extract abundant attributes, which describe the concepts, and the semantic relationships between attributes. The most efficient and effective Ontology technique of detecting the semantic relation between words is the WordNet. We extend each attribute into a concept set which is used for semantic matching.

In the previous work, attributes of the query interfaces were obtained manually and the co-occurrence of attributes was used to evaluate the domain information [6]. In our framework, we measure the relevance of attributes not only with the exact matching, but also with the semantic similarity. The automatic attribute extraction is the indispensable previous pretreatment of schema matching, schema merging, and the carrying out of some correlated research fields depended on the result of it, such as discovering, categorizing, indexing, and query capability modeling Deep Web sources and extracting domain knowledge [7-9]. However, the automatic attribute extraction has always been proven to be a difficult task [10]. Yoo, et al. have put forward an automatic attribute extraction algorithm to automatically determine the attributes of Deep Web query interfaces by utilizing WordNet[11]. Two types of attributes, programmer viewpoint attributes and user viewpoint attributes, were defined. The final attributes of a query interface were determined by checking the overlapping areas between programmer and user viewpoint attribute sets. The most

Corresponding author: WanLi Zuo, Wanli@jlu.edu.cn

doi:10.4304/jsw.5.7.713-720

obvious difference between the methods of Yoo's and ours is that we extracted the instances information during the attributes extraction procedure. We believed instances are a major part of describing the semantic attributes and sometimes provide instantial information in query interfaces to depict the meaningless words, such as "From", "To", "ISSN", et al. The semantic of query interfaces is not complete with some meaningless words semantically un-instantiated. The recall of query translation is lowered by the semantic incomplete query instances.

In this paper, we present a translation framework of translating queries utilizing automatic attributes extraction and Ontology technique. During the procedure, the instance information is extracted and used to instantiate meaningless attributes and semantic containers based query translation mechanism reconcile the semantic relationships between attributes. Ontology provides semantic support by using controlled terms for attributes mapping in a domain. We combine query interfaces of the Deep Web and Ontology approach of the Semantic Web. Summing up the previous works, our approach makes the following contributions:

1. The query interface attributes are automatically extracted from query interfaces and the instances information are used to instantiate some attributes semantically. By instantiating meaningless attributes, the semantic completeness is confirmed. During attributes extraction, Ontology technology is utilized to extend the pre-matching attributes, in order to ensure the effectiveness of attributes matching between different query interfaces of Deep Web. We improve the recall and precision of attribute extraction by using the instance information contained in query interfaces.
2. The precision of queries translation is improved by the translation mechanism based on the extracted semantic container, by which query interfaces semantic restrictions are described.

II. AUTOMATIC ATTRIBUTES EXTRACTION ALGORITHM

In Fig 1, we can see that all the information of the query interface showing in the form of HTML. The HTML shows a number of common elements. The <label>, <select>, and <option> occur in pair style, and there are some element names and texts showing between each pair of elements. Take the label "Depart From" for example, the "departure city" is the element name given by the programmer in the programming environment, and "Depart From" is text attribute of the element. When users query against the query interface, they can only see text attributes of the form elements, but after users pushing down the search button the translation between the name and text of the element is carried out. So we take EN short for element name and ET for element text. Both EN and ET are important information to get the semantic understanding of a query interface. In our demo, we extract EN set and ET set of a query interface to determine valid attributes, candidate attributes are stored in the refined EN set and ET set. Given a set of query interfaces, the EN set are obtained from the inner identifiers, the ET set are obtained from free text within the query interface.

In Fig 2, the candidate automatic attributes extraction algorithm is composed of three steps: extracting EN set; extracting ET set and generating valid attributes. During the ET extracting, the instance information is extracted together and attached to semantic related candidate attributes. By attaching instance information, some meaningless words are instantiated by instances. This process is very necessary to deal with situations of query interfaces semantic incompleteness.

```
<FORM action= "..." method= "…">
<P><LABEL for= "departure city">Depart from <BR></LABEL>
<SELECT size= "2" name= "depart_city">
<OPTION selected value= "city1">New York</OPTION>
<OPTION>Washington</OPTION></SELECT><P>
Where is your departure from? <BR>
<INPUT type= "text" id= "id"><P> Search by :< BR>
<INPUT type= "radio" name="search by" value="only"> only<BR>
<INPUT type="radio" name="search by" value=
"schedule">schedule<P>
```

Figure 1. HTML code of query interface

Algorithm: Candidate Attribute Automatic Extracting

step 1. Get IIS (inner identifier set) and TIKW from web data source,
step 2. Remove special symbols, generate more substrings. Special symbols (:, -, _, @, $, &, #, ?, !, *,etc.)
step 3. Remove duplicated in the IIS and TIKW.
step 4. Extract all text between <option> and </option> form the instance set, INS for short.
step 5. Pre-process function, PPF for short.
step 6. Extend the key words of IIS, TIKW and INS into a set by utilizing WordNet.
step 7. Generate hierarchy tree for EN and ET.
step 8. Add instances into ET tree mark the relation between candidate attributes and instances; mark the relation between candidate attributes and instances.
step 9. Refine the hierarchy tree and get EN and ET set in a hierarchy relation tree.

Figure 2. Algorithm of candidate attribute automatic extraction

A. EN extraction algorithm

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 preprocess function step is used to finish these preprocessing tasks. First, there are some information retrieval pre-processing method should be used, such as stop word removal and stemming. Some words we get from query interface are no value, so removing stop word and stemming attribute names and domain values. We can build domain stop word list manually according to general stop word list. Second, we have to expand 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. Last, during this step, 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", and "first-name" into "first name". However, this step has to be done with care. For example, we may have "Baths" and "Bathrooms" if we split "Bath" with "rooms" it could be a mistake because "rooms" could have many and different meanings. To avoid this situation, we need to make sure that "Bathroom" is not an English word before splitting it, so we need an English dictionary or domain describing taxonomy.

Ontology is employed for processing text from the query interfaces of Deep Web sources efficiently filtering words from the Deep Web data sources. The Ontology adds a semantic layer to the Deep Web. In this paper, WordNet is utilized as a kind of Ontology instrument for extending the candidate key words and finding matches between EN set and ET set, based on synonyms. It is also used for eliminating stop words to allow correct attribute retrieval. Among the WordNet categories, nouns, verbs, adjectives, and adverbs, we focus on nouns based on the observation that semantics are mostly carried by them.

Take the key words set S_1= {departure city, airline code} as an example, we extend the each word and keep their semantic relation together by using WordNet. In Fig 3a, the "departure city" is extended into $E_{departure}$= {departing, going away} and E_{city} = {metropolis, urban center}. The hierarchy tree is generated by extending procedure. Sometimes in the query interface, there are so many candidate key words for the inner identifier, so we should refine the hierarchy tree to get more general candidate attributes and specific semantic relations. The refined result shows in Fig 3b. During the refining step, we get more general words for the candidate key words and delete the phrase. The phrase expression is difficult to match between each other and it has more complicated meanings.

B. ET extraction algorithm

ET sets are utilized to determine the final attributes of each Web data source too, and they are obtained from the free text within the query interface. It requires that the free text between two HTML tags, which potentially embodies semantics, is added into the set TIKW (text-based identifier key words). Texts between <option> and </option> are also considered because they describe instances for candidate key words. We believe that the instances contain semantic information which describing the attribute, and we can take the attribute as the concept of Ontology. So we try to extract the instances in the text section which is not like Yoo's, because they thought the instance was no value to be concerned during attributes extraction. We finish four different parts of extraction work. First, we generate the text-based identifier key words. Second, the instance information can be extracted from the text of the query interface. Thirdly, after getting valid candidate attributes, we generate the hierarchy tree for the ET. Finally, add instances into the ET tree and mark the relation.

In order to add right instances into the ET tree, we need to design a method to measure the semantic relation between the instance and the candidate attribute. In the query interface, the layout format includes the information describing the relation between the each labels and elements and free texts of the query interface. We find that there are two heuristics info can be calculated to measure the semantic relation between the correlative elements of the query interface.

C. Finding semantic relation between text instance and candidate attribute

At this step, we consider the texts of instance information in query interface and compute the visual distance with respect to each field of the form F. The visual distance between a text instance *ti* and a candidate attribute *ca* is computed as following:

1. We can use the APIs provided by the browser to obtain the coordinates of a rectangle enclosing *ca* and a rectangle enclosing *ti*. If *ti* is in a HTML table cell, and it is the unique text inside, then we can mark the correlative relation between them.
2. We also obtain the angle of the shortest line joining both rectangles. The angle is approximated to the nearest multiple of $\pi/4$.

For each candidate attribute, we try to obtain the texts instance semantically linked with it in the query interface. For selecting the best matching text information for a *ca*, we apply the following steps:

1. We add all the text instances with shortest distance with respect to ca into a list.
2. Those text instances having a distance lesser with respect to ca are added to the list ordered by distance. This step discards those text instances that are significantly further from the field.
3. Text instances with the same distance are ordered according to its angle. The preference order for angles privileges texts aligned with the fields. The main standards to measure the preference is that privileges left with respect to right and top with respect to bottom, because they are the preferred positions for labels in forms.
4. After extracting the ET, we also can get a tree hierarchy exhibition of the candidate attributes. In the Figure 3c, it shows an ET tree extracted from the HTML of query interface showed in Fig 1. In the Fig 3d, it shows the refined ET tree and added instances.

In the Fig 3c, the "from" and "search" are in dashed blocks, because they are a little different from the other terms. We can get no information form WordNet about "from" and we also find that "search" is the more general word in the group of the same meaning words set. In this situation, "from" and "search" are called atom lemmas. So there are no extra information form the WordNet and they have no leave nodes comparing with "depart" and "by".

Definition 1: In WordNet, given a word w_1 is in the higher level of the synset or there is no description information about w_1, w_1 is called atom lemma.

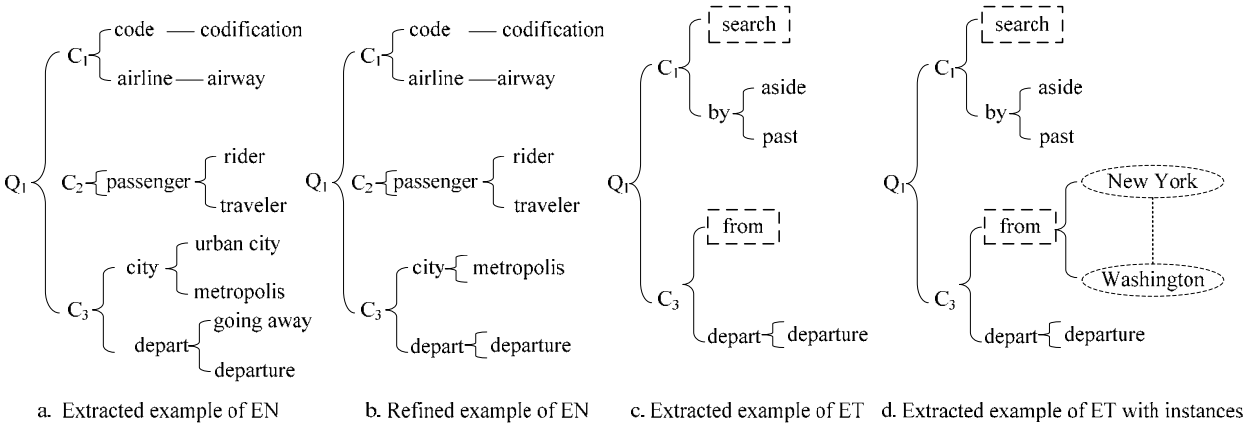
Definition 2: In WordNet, given word w_1 and w_2, if there exits "w_2 IS INSTANCE OF => w_1" relationship between two words, then w_1 is called source lemma and w_2 is called instance lemma of w_1.

In the Fig 3d, there are two text instances and showed in circle blocks. We believe that the instances information

are belonged to both "depart" and "from". It is easy to find that the instance "New York is an INSTANCE OF city, metropolis" and "urban center" in the WordNet. In other words, "city" is the source lemma of "New York". We describe the relation between candidate attributes and instance like:

S_{depart}= {depart || departure, INSTANCE ((New York, "instance of" (city, metropolis)), (Washington, "instance of" (city, metropolis)))},

S_{from}= {from || INSTANCE ((New York, "instance of" (city, metropolis)), (Washington, "instance of" (city, metropolis)))}.



a. Extracted example of EN b. Refined example of EN c. Extracted example of ET d. Extracted example of ET with instances

Figure 3. Examples of algorithm

The instance information and candidate attributes is expressed by hierarchy relation during the programming for the framework. We name the 'depart' as the value of S_{depart} and the "departure" as alias. The next step of our framework is to generate the finial valid attributes.

D. Final Attribute extraction algorithm

After extracting the EN set and ET set, we can build the candidate attributes mapping between the two sets and get the final attribute set. The procedure of the final attributes extraction is to merge the two trees of EN set and ET set. In the tree, we don't break the semantic relation between the two words shown together, such like "depart from" and "departure city", because in human's sense they express the name of passengers' start point of the journey. The programmers and users of the query interface are in the same human sense, so we believe it is necessary to keep the semantic relation.

Before describing the final attribute extraction algorithm, we explain the different kind nodes of the EN and ET tree. There are four kinds of node in the tree:

- **Instance node**: the instance extracted from the free text, such as "New York" in Fig 3d. The instance nodes are always shown in the bottom of the tree, so *L* (leaf) is short for the instance node.
- **Ally node:** the extended words for the key words derived from WordNet, such as "departure" is an ally node of "depart". *A* is short for the ally node.
- **Value node:** the extracted key word from the query interface, such as "depart". *V* is short for the value node, $V = \{v \mid a_1, a_2 \ldots a_n, I(l_1), I(l_2) \ldots I(l_m)\}$.
- **Semantic node:** the direct ancestor node of value node, there are at least one value node below the semantic node. The semantic node is the set of related value nodes. *S* is short for the semantic node, $S = \{v_1, v_2 \ldots v_n\}$.
- **Root node:** the root of ET or EN tree.

We take the ET tree as the blueprint and EN tree as the source tree, merge the candidate attributes of EN into ET. The final attribute extraction is composed of three steps. First, we generate the ally nodes by instance node. During this procedure, we get the hypernyms of the instance node from WordNet and create ally nodes using them. After creating ally nodes, we need attach the new ones to value nodes which the instance nodes belonged to. There are two situations to be concerned, one is that there are ally nodes belong to the value node and the other is there is no ally node belongs to the value node. In the first situation, we will not attach the new ally nodes to the value node, because these kinds of value nodes have been extended by the WordNet. In the last situation, we attach the new ally nodes to the value nodes directly to improve the precision of matching, because these kinds of value nodes are always meaningless words or the most general words. Second, we merge the semantic nodes. During this step, we merge the semantic nodes when, at least, one of the two sets of value nodes is matched. After merging, we get the tree of ET', in it the semantic nodes are expanded. Third, we merge the value nodes. We rearrange the value nodes and the ally nodes, delete the reduplicate ones and sometimes exchange the value nodes and ally nodes. The rule of the exchanging is basing on the ET tree and making the more general words in the value node. The algorithm is shown in Fig 4.

E. Generating semantic container from query interface

We observe that Deep Web sources often have two types of constraints on how predicate templates can be queried together. First, a form may have binding constraints, which require certain predicate templates be filled as mandatory. For instance, the form in Figure 6a may require price not be queried alone and thus each form query must bind a predicate template from attributes. Second, some predicate templates may only be queried exclusively. For instance, the form in Fig 6b

allows only an exclusive selection among attributes "Writer", "Title" and "ISBN". To solve this kind of problems, we need to find relation between attributes in the source and target query interface with the help of semantic meanings of the web. We take the query interface in Fig 5a as the source query interface and the one in the Fig 5b as the target query interface.

Algorithm: Final Attribute Extraction (*EN, ET*)

*/*generate instances for value nodes in EN and ET*/*
1. For each *value node v* in EN and ET do
2. If *v* has instance
3. Generate ally nodes for *v*

*/*merge semantic related nodes in EN and ET*/*
4. For each sn_i do (sn_i *is a semantic node in EN tree,* sn_j *is a semantic node in ET tree)*
5. Find similarity between two *semantic node sets*
   IF exits *semantic nodes* matched
6. Merge *sni* into *snj*
7. New generated *semantic node* in ET is marked
8. For each *semantic node sn* in *ET* do
9. If *sn* not marked
10. Remove *sn* from *ET* and generate *ET'*for *ET*

/* generate final attribute tree */
11. For each *semantic node sn* in *ET'* do
12. For each *value nodes v* in *sn* do
13. Find similarity between each pair of *value nodes* in *ET'*
14. Merge value nodes
15. Return *ET'* as the final attribute tree

Figure 4. Final attribute extraction algorithm

Definition 3: Given a set of attributes of a query interface, if a sub-set and only one attribute in it can be translated with other attributes into a valid query condition, then the relation between the attributes in it is called exclusive.

Definition 4: Given a set of attributes of a query interface , if a sub-set and all the attributes in it can be and must be translated together into a valid query condition with the semantic constraint provided by the query interface, then the relation between the attributes is called binding.

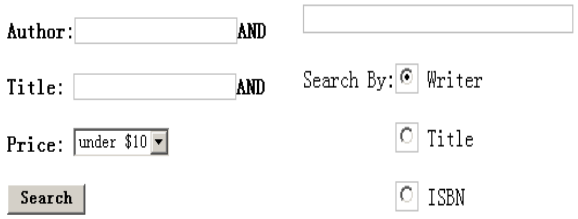
In Fig 5b, the function of the form is to provide queries which contain only one of "Writer", "Title", "Subject" and "ISBN". The four query terms form a set, each time we can only query about one of them. It is necessary to find the relation between them when translating the query for the target. It is obviously that we can find some evidences from the code of the target form. It is easy to find that the "Writer", "Title" and "ISBN" share the same container named 'RB1' from the code. We define a predicate container to present the attributes being constrained by the web semantics defined by the author of the web form.

Definition 5: The semantic container is defined as $C = \{a_1, a_2, \cdots a_n \cong relation\}$, a_i presents the attributes in the same container, "relation" presents the relation between the attributes.

$$Csource = \{\text{Author}, \text{Title}, \text{Price} \cong \text{binding}\}$$

$$Ctarget = \{\text{Writer}, \text{Title}, \text{ISBN} \cong \text{exclusive}\}.$$

After automatic 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 $P =< attribute, constr, value >$. Taking the attribute "Author" as an example, the predicate template is <*author, like, 'Joanne Kathleen Rowling'* >. It is a kind of text template, which it is the most frequently template used in the Deep Web. 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 interface, user can use three 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 interface, user can use one of all the attributes to describe one facet of the book each time. To get translation of the different query interfaces, we have to get more valid attributes as we can. If we have some domain knowledge about book, we will find the "price" is the least important attribute when describing a type of book, there are the same situations in the other domains. When translating queries, it is better to make numeric attributes useless.



a. Source query interface b. Target query interface

Figure 5. Query interfaces with different semantic constraint

□.RESULTS OF EXPERIMENTS AND DISCUSSION

The Deep Web data sources were downloaded from the UIUC web integration repository. This dataset contains original query interfaces and their manually extracted query capabilities of 447 Deep Web sources. We first extract the attributes in different domains to test our automatic attributes extraction method. Take Airfares and Automobiles as examples, we get the results in the Table 1. The parameters are described as following:

- CA: candidate attribute;
- EA: extraction attribute;
- IA: instance attribute;
- $I / E = |IA| / |EA|$;
- $E / C = |EA| / |CA|$.

We can get three clustered classes of Deep Web query interfaces. The one of the clustered classes is characterized by that there is none instance attribute in the form. There are more than one instance attribute in the second class of interfaces. The last class reflects both strengths and weaknesses of our extraction algorithm, just like showing both sides of the coin. The first side is that our algorithm can find the instanced candidate attributes in the very complicated coded query interfaces, while we can not extract anymore attributes. The other side is that when extracting the attribute embedded in a longer and

irregular string, our algorithm shows inefficient. In Ref [11], they removed all the information contained in the instance between <option> and </option>, but they had difficult in making the meaningless words meaningful in the semantic query interface, such as "from", "to", .etc. It is true that the number of meaningless words is very limited in the interfaces of the Deep Web and we can manually deal with the problem by building some related mappings, but we believe that the automatic extraction is very necessary. The Deep Web is heterogeneous and grows rapidly, so we can't foresee the future new emerging domains of people life just like we can't reach all the Deep Web. The work we have done is just like adding some neologies such as "blog", "newbie", "Wiki", .etc, into a dictionary by making them meaningful in some domains.

TABLE I. THE RESULTS OF EXTRACTION IN AIRFARES AND AUTOMOBILES DOMAIN

WWW	CA	EA	IA	E/C(%)	I/E(%)	WWW	CA	EA	IA	E/C(%)	I/E(%)
Airfares											
spiritair.com	12	2	2	16.7	100	united.com	15	12	2	80.0	16.7
usairways.com	12	2	2	16.7	100	airfrance.usc	14	14	2	100.0	14.3
skyauction.com	5	5	3	100.0	60.0	hotwire.com	7	7	1	100.0	14.3
aireuropa.com	8	4	2	50.0	50.0	flyairnorth.com	20	19	2	95.0	10.5
flyasiana.com	10	10	4	100.0	40.0	flights.com	19	16	1	84.2	6.3
southwest.com	12	11	3	91.7	27.3	bargaintravel.com	11	9	0	81.8	0.0
priceline.com	10	8	2	80.0	25.0	cheapairlines.com	8	8	0	100.0	0.0
134.americanexpress.com	11	9	2	81.8	22.2	delta.com	13	10	0	76.9	0.0
lowestfare.com	9	9	2	100.0	22.2	expedia.com	11	9	0	81.8	0.0
continental.com	13	10	2	76.9	20.0	faremax.com	12	8	0	66.7	0.0
finnair.com	10	10	2	100.0	20.0	nwa.com	13	12	0	92.3	0.0
koreanair.com	10	10	2	100.0	20.0	orbitz.com	16	12	0	75.0	0.0
aircanada.com	18	16	3	88.9	18.8	res99.com	8	7	0	87.5	0.0
singaporeair.com	11	11	2	100.0	18.2	smartertravel.com	7	7	0	100.0	0.0
Automobiles											
classicmo.com	23	16	3	69.6	18.8	inventory	32	24	2	75.0	8.3
drive.com	20	18	3	90.0	16.7	carsite.com	14	12	1	85.7	8.3
cunninghamgm	22	19	3	86.4	15.8	gmgiant.com	15	13	1	86.7	7.7
motors.search.ebay	29	22	3	75.9	13.6	carsforsale.com	14	13	1	92.9	7.7
inventory_default	33	25	3	75.8	12.0	xtra.autopoint	15	13	1	86.7	7.7
autobytel.com	23	17	2	73.9	11.8	barriermotors	15	14	1	93.3	7.1
autoweb.com	20	17	2	85.0	11.8	autofinder.com	17	14	1	82.4	7.1
2buycars.net	21	18	2	85.7	11.1	audi.traverse	17	14	1	82.4	7.1
cars.com	20	18	2	90.0	11.1	akguam.com	20	15	1	75.0	6.7
carsdirect.com	22	18	2	81.8	11.1	tonkin.com	22	16	1	72.7	6.3
gotcars4sale.com	10	10	1	100.0	10.0	inventory	11	10	0	90.9	0.0
kbb.com	10	10	1	100.0	10.0	autobroker.net	6	6	0	100.0	0.0
autonetmail.com	25	21	2	84.0	9.5	autonation.com	4	4	0	100.0	0.0
autos.msn.com	13	11	1	84.6	9.1	autoseek.com	9	8	0	88.9	0.0
lead.carprices.com	12	11	1	91.7	9.1	worldparts.com	5	5	0	100.0	0.0

The second step of our experiment is to translate queries from one query interface to another in the same domain. In this paper, we carry out experiment on Airfares, Automobiles, Books, Music and Movies domains. The experiment results and the comparison with Yoo's method are shown in Table 2. The parameters are defined as following:

- QN: the number of query interfaces in specific domain;
- AEP: the average precision of attribute extraction in specific domain;
- TR: the recall of query translation in specific domain;
- TP: the precision of query translation in specific domain;
- Yoo-TP: the precision of query translation of Yoo's algorithm.

The translating precision of our method is 2% higher than Yoo's on average. Especially in the Airfares domain, we take full advantage of the text instance information to improve the attribute extraction precision and translating precision. However, there are some lacks of our method in the Books and Music domains. The semantic container is difficult to capture in the two domains and the default

processing of the semantic container is not effective enough.

TABLE II. THE RESULTS OF AIRFARES, AUTOMOBILES, BOOKS, MUSIC AND MOVIES DOMAINS

Domain	QN	AEP	TR	TP	Yoo-TP
Airfares	35	85.3%	81.7%	76.7%	67.8%
Automobiles	36	87.3%	82%	74.5%	70%
Books	25	93%	85%	79%	100%
Music	25	87%	83%	72.2%	77%
Movies	25	95%	87%	82.7%	60%
Average:	29	89.5%	83.7%	77%	75%

During the experiment we find that:

1. It is not necessary to fill all the form controls. There are different and special attributes in each query interfaces. In order to get related results from different query interfaces, we need only translate the query instances on the common attributes of the query interfaces.
2. Some complex form controls and semantic relation are too hard to deal with. In the situation of absenting semantic container, the default semantic relation between the attributes is binding, which can make sure a high recall of related results.
3. Before carrying out experiments, we are assuming that Web programmers are well educated and are using meaningful field names, but it is not the truth. The programmers of different region use different meaningful field names and the programmers of the same region also have different field named mechanism.
4. Some forms are designed under the universal templates and the attribute named mechanisms are under the team instruction, so some attributes are expressed in long and irregular strings of characters. Some special characters, such as "#", "$"and "￥", are also used in the strings. The large-scale use of advanced graphics controls and uncommon design method also lower the precision of the attributes extraction.

There is still something we can do to improve the precision of translation.

.FUTURE WORK

Our framework illustrates an automatic process of extracting attributes from query interfaces and query translating. We propose to get more valid attributes by using Ontology technology, each candidate attribute is extended into form a synonym set by WordNet and stored in a tree data structure. During the attributes extracting, the instance information is extracted together in the aim of being attached to semantic attributes and instantiating meaningless attributes. As we all know, attributes in the query interface are still part of the schema hidden in the Web database, so in our opinion the semantic completeness of the query interface is very important factor of determining precision and recall of query translation. After attributes extraction, query translation is carried out based on the semantic containers generated from query interfaces. The precision and recall of query translation are improved by attributes extracted and instantiated by our algorithm, especially in the Airfares domain. However, the precision and recall are not effective enough, because the semantic restrictions in query interfaces are too hard to deal with only by parsing technique. The design style and purpose of query interfaces can not be modeled by anyone else except the designer.

ACKNOWLEDGMENT

The research is under the support of Natural Science Foundation of China under grant No. 60973040, 60903098; the Specialized Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 200801830021; the Science and Technology Development Program of Jilin Province of China under Grant No.20070533; the Basic Scientific Research Foundation for the Interdisciplinary Research and Innovation Project of Jilin University under Grant No.200810025.

REFERENCES

[1] Bin He, Kevin Chen-Chuan Chang. Statistical schema matching across Web query interfaces. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June, 2003, pp: 217-228.

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

[3] B. He, Zhen Zhang, and Kevin C.-C. Chang, MetaQuerier: Querying Structured Web Sources On-the-fly, In Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June, 2005, pp: 927-929.

[4] B. He, K. C.-C. Chang, Automatic complex schema matching across web query interfaces: A correlation mining approach, ACM Transactions on Database Systems, Association for Computing Machinery, Vol.31(1), 2006, pp: 346-395.

[5] B. He, Patel M, Zhang Z, K. C.-C. Chang, Accessing the deep web, Communication OF The ACM, Vol.50 (5), MAY 2007: 95-101.

[6] G. Kabra, C. Li and K.C. Chang. Query Routing: Finding Ways in the Maze of the Deep Web. In Proceedings of the International Workshop on challenges in Web Information Retrieval and Integration, Tokyo, Japan, April, 2005, pp: 64-73.

[7] Caverlee J, Liu L, Rocco D, Discovering Interesting Relationships among Deep Web Databases: A Source-Biased Approach, World Wide Web-Internet and Web Information Systems, Vol.9(4): 585-622, Springer, 2006.

[8] Shu LC, Meng WY, He H, Yu C, Querying Capability Modeling and Construction of Deep Web Sources, In Proceedings of 8th International Conference on Web Information Systems Engineering, Nancy, France, Dec, 2007, pp: 13-25.

[9] LIU Wei, MENG Xiao-Feng, MENG Wei-Yi, A Survey of Deep Web Data Integration. Chinese Journal of Computers, Vol.30, No. 9, Sept, 2007, pp: 1475-1489.

[10] Sriram Raghavan, Hector Garcia-Molina. Crawling the hidden Web. In Proceedings of 27th International Conference on Very Large Data Bases, Roma, Italy, Morgan Kaufmann, September, 2001, pp: 129-138.

[11] Yoo Jung An, James Geller, Yi-Ta Wu, Soon Ae Chun: Semantic deep web: automatic attribute extraction from the deep web data sources. In Proceedings of the 2007 ACM Symposium on Applied Computing (SAC2007), Seoul, Korea, March, 2007, pp: 1667-1672.

LIANG Hao was born in Jilin of China in Jun 1980. He is a Ph.D. candidate at the Department of Computer Science and technology, Jilin University. His current research interests include Web Intelligence, Ontology Engineering and Information integration.

ZUO Wan-Li was born in Jilin of China in Dec 1957. He is a professor and doctoral supervisor at Department of Computer Science and technology, Jilin University. Main research area covers Database Theory, Machine Learning, Data Mining and Web Mining, Web Search Engines, Web Intelligence.

He was as a senior visiting scholar, engaged in collaborative research in Louisiana State University (USA) from 1996-1997. He was principle responsible member of 3 national NSFC programs. More than 40 papers of him were published in key Chinese journals or international conferences, 8 of which are cited by SCI/EI. Three books were published by him in Higher Education Press of China and he obtained 3 national and departmental awards.

He is a member of System Software Committee of China's Computer Federation, prominent young and middle-aged specialist of Jilin Province.

Ren Fei was born in Inner Mongolia of China in Aug 1981. She is a Systems Architecture Engineer in China Development Bank. She received Ph.D. at the Department of Computer Science and technology, Jilin University. Her current research interests include Data Mining, Intrusion Detection, Information Security, and Artificial Immune Algorithm.

FengLing He was born in Jilin of China in May 1962. He is a professor and supervisor at Department of Computer Science and technology, Jilin University. Main research area covers Data Mining and Web Mining, Web Search Engines, Web Intelligence.