Article ID:

Query Translation on the Fly in Deep Web Integration

Jiang Fangjiao, Jia Linlin, Meng Xiaofeng*
Information School, Renmin University of China, Beijing 100872, China

Abstract: To facilitate users to access the desired information, many researches have dedicated to the Deep Web (i.e. Web databases) integration. We focus on query translation which is an important part of the Deep Web integration. Our aim is to construct automatically a set of constraints mapping rules so that the system can translate the query from the integrated interface to the Web database interfaces based on them. In this paper, we construct a concept hierarchy for the attributes of the query interfaces, especially, store the synonyms and the types (e.g. Number, Text, etc.) for every concept. At the same time, we construct the data hierarchies for some concepts if necessary. Then we present an algorithm to generate the constraint mapping rules based on these hierarchies. The approach is suitable for the scalability of such application and can be extended easily from one domain to another for its domain independent feature. The results of experiment show its effectiveness and efficiency.

Key words: Deep Web data integration; query translation

CLC number: TP 393.09

0. Introduction

In recent years, users could access the interesting information from Deep Web (i.e. Web databases) which has been developed rapidly. As reported [1], there are 96,000 web sites and 550 billion hidden pages in the Deep Web, which is 500 times more than the Surface Web. To help the users find the desired information from the Deep Web, many researchers carried out their research works on the Deep Web integration. But these researches mainly focus on interface integration and result page extraction, some of which address the Web databases crawling, discovery and classification. A few efforts are dedicated to query translation which has responsibility for translating a user's query from the integrated interface to the web database interfaces.

Research works related to query translation have mainly fallen into two categories: attribute mapping and constraints mapping.

Attribute mapping [2, 3, 4, 5, 6, 7] in Deep Web integration has been extensively researched. These works can be classified into two different methods, one [2,3] is schema-based method, and the other [4,5] is instance-based method. The paper [2] takes a conceptually novel approach by viewing schema matching as correlation mining and proposed a new correlation measure, H-measure, to find the mapping attributes. The paper [3] utilizes statistics technology other than data mining one. The paper [4] proposes an interactive, clustering-based approach and the paper [5] proposes a data-ensemble framework with sampling and voting techniques, respectively. Instance-based methods have been employed in many schema matching tasks [6,7]. The paper [6] addresses two significant schema matching problems: intra-site and inter-site. WebIQ[7] proposes a solution that learns from both the Surface Web and the Deep Web to automatically discover instances for interface attributes. Representational research works on constraints mapping are [8] and [9]. The paper [8] applies user provided mapping rules to translate query. In the paper [9], the approach dynamically mapping predicates across unseen sources is proposed.

Due to the autonomous, heterogeneous, dynamic and scalable nature of the Deep Web, query translation will be a

Received date:

Foundation item: Supported by the Natural Science Foundation of China (60573091); the Natural Science Foundation of Beijing(4073035); the Key Project of Ministry of Education of China (03044).

Biography: Jiang Fangjiao (1971-), Female, Ph.D. candidate, research direction: Web data integration. Email: jiangfj@gmail.com, Tel: 010-62512334

* To whom correspondence should be addressed. Email :xfmeng@ruc.edu.cn

complex and challenged key point.

In this paper, we want to address the problem of how to automatically translate a user's query from the integrated interface to a series of Web database interfaces. In details, our contributions are follows.

1. We propose two kinds of hierarchy relationships, concept hierarchy and data hierarchy to aid to translate query.

2. We propose an effective algorithm based on the two kinds of hierarchy relationships for generating query translation rules automatically. It consists of two parts which are simple query translation and complex query translation.

3. The results of the experiment show our algorithm is both effective and efficient.

1. The problem of query translation

Definition (**Query Translation**): In Deep Web integrated system, given a query Qi in the integrated interface and a random Web database interface in the same domain, suppose the Q* on the exact Web database interface is an effective and valid query correspondent to the Qi, (i.e. the Boolean constraints of Q* is equal to or subsume those of Qi as minimal as possible.) The process that translates the Qi to Q* is named as **query translation**.

Given an integrated interface in one domain, the problem is how to automatically translate the query to the Web database interfaces. First, we should find out the attribute matching relationships between the integrated interface and the Web database interfaces. Then, we will consider the value translation of mapped attributes.

The integrated interface usually contains the most important and detailed attributes which are selected from a set of Web database interfaces and organized in the way that is more convenient to users to understand. But the schemas of the Web Database interfaces are diversified due to the autonomous and heterogeneous nature. Especially, it is very difficult to find the correct mapping attributes when the Web database interface is changed or it is a new one. The attribute matching relationships between the integrated interface and the Web Database interfaces are complex as follows:

1. The complex attribute mappings are common. e.g. {Month, day, year} = {date}.

2. The names of attributes belong to the same concept are always different, e.g. attribute name “departure city” and “leaving from”.

3. Some attributes look like synonyms, but in fact, they belong to different concept. e.g. “departure city” and “from” look like synonyms, but the latter represents the leaving time in some Web database interfaces.

In addition, some problems exist in the data translation of the mapped attributes:

1. In complex attribute mapping, the data of attribute represent the higher level concept always consists of several attribute data.

2. Even in simple attribute mapping, the data inconsistency will lead to complex data translation, e.g. “leaving time=9:00am” in the integrated interface while the list value of “leaving time” of a Web database interface is “8:00am” or “12:00am”.

This is easy for human user to find the semantic mapping relationship and values corresponding relationship of the attributes. But translating query manually will limit the scalability of such application, so it is critical to propose an approach to fulfill this task automatically.

2 The Framework of query translation

According to above analysis, our solution consists of two parts. First, we create two kinds of hierarchy relationships, concept hierarchy and data hierarchy. And then, based on them, we propose an effective algorithm to automatically generate query translation rules to process not only the simple (1:1) data translation but also complex (m:1) data translation. Our query translation framework is showed in Fig.1.

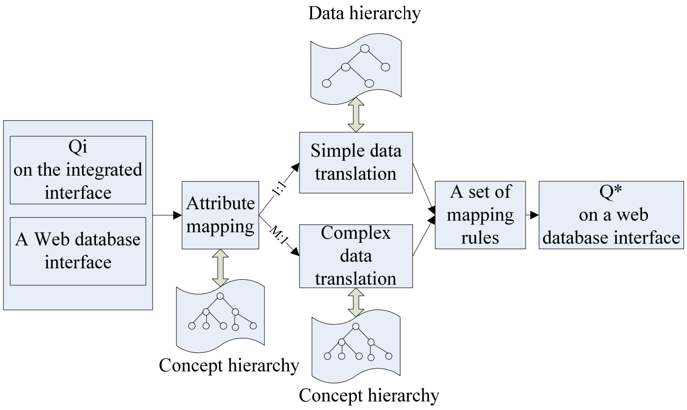


Fig. 1. The processing flow of query translation

2.1 Hierarchy construction

Concept hierarchy records the subordinate or synonymous relationships and the types of the attributes of the Web database interfaces in the same domain and data hierarchies express the relationship between the values of same attribute.

2.1.1 Concept hierarchy

Some studies focus on constructing an integrated interface for each domain based on these semantic relationships. Furthermore, hierarchy relationships[10] of the attributes in the integrated interface have been constructed which group the related pieces of information together to express high level concept (e.g. group “first name” and “last name” express “author”). It is good enough to facilitate users to understand and fill explicit information. But it is not enough to map the attribute from the integrated interface to any Web database interface because of its autonomous and dynamic nature.

To address this problem, synonyms of the attribute collected from Web database interfaces should also be stored in the relationship. And according to the survey, the attributes distribution in one domain obey Zipf’s law, so we can consider the attribute names that appear frequently in Web database interfaces and store them with respect to the attribute synonym relationship between the attributes in the integrated interface and those in the Web database interface.

In addition, during the attribute mapping process, we have observed that the types of attributes are very important information. For example, in airfares domain, attribute “from” in one Web database interface represents the departure site, while in another one it represents the departure time. Although it is difficult to distinguish its meaning only according to its name, it is much easier to tell what is actually expressed aided by its data type. In above example, the attribute “from” with text type represents the departure site, while with date time type represents departure time. So we store the type in the concept hierarchy as well as attribute name. In our approach, we consider the basic data types: Date time, Numeric and Text.

Definition (**Concept hierarchy**): A concept hierarchy is a directed tree consisting of a set of nodes N. Each node represents a concept which is defined as Node= (K, DT, {Si}, {Li}). It contains the following information.

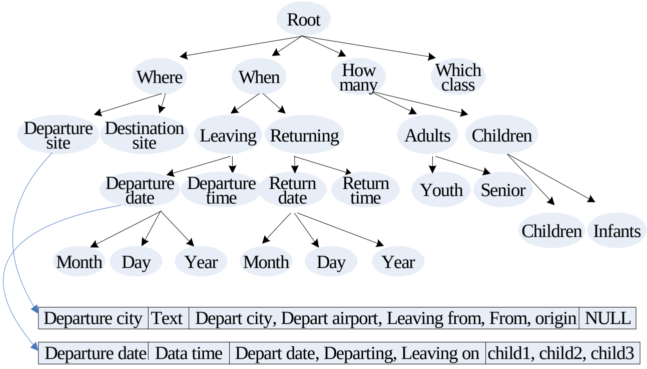
K A keyword which represents a concept

DT Data Type which the concept belongs to

{Si} A number of synonyms of the keyword K

{Li} A number of links pointing to its child nodes

The concept represented by the child node is usually a member of or a part of the one represented by its parent node.



Departure city	Text	Depart city, Depart airport, Leaving from, From, origin	NULL
Departure date	Data time	Depart date, Departing, Leaving on	child1, child2, child3

Fig. 2. an example of concept hierarchy

During constructing the concept hierarchy, the K can be found out by method proposed in [10], the DT can be analyzed through parsing HTML form-based query interfaces, and {Si} or {Li} can be access from existing scheme matching works on the Web database interfaces. Fig.2. shows an example of a concept hierarchy in airfares domain.

The synonyms are used to attribute mapping, while data types are used to data translation as well as attribute mapping. The corresponding algorithm will be described in following sections.

2.1.2. Data hierarchy

In some cases, two attributes belong to the same concept (e.g. departure city and departure airports both represent the departure site), but there are not correspondent values. The values exists contained, overlapped or similar relationship. It always leads to complex data translation.

When the data type is Numeric or Data Time, the most appropriate value can be easily found out by projecting the original value to the numeric or time axis. If the data type is Text, the case will be more complex, because the contained, overlapped or similar relationships between the texts are fuzzy. We construct the data hierarchy manually for some attributes belong to the same concept to make the relationships of the attribute values more explicit. For example, the data hierarchy on the departure site is constructed as Fig.3.

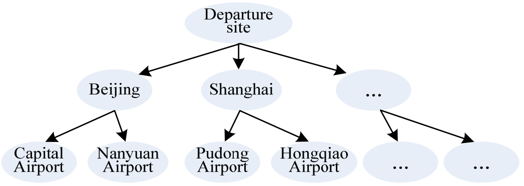


Fig.3. an example of data hierarchy

2.2. Query translation algorithm

We try to automatically generate the mapping rules to suitable for the autonomous and dynamic nature of query translation in the Deep Web integration application. The generation of the mapping rules should be considered from two aspects: Attribute mapping and Data translation.

2.2.1 Attribute mapping algorithm

Suppose that $Attr_i$* represents an attribute name of a Web database interface, $Attr_j$ represents the attribute concept of a node N_j in the concept hierarchy and $Attr_{ju}$ is a keyword or any synonym of node N_j. If the similarity of $Attr_i$* and $Attr_j$ is over a predefined threshold, we say that $Attr_i$* is mapped to N_j.

Definite (similarity):

$$Sim(Attr_i^*, Attr_i) = w_t \times (Sim_t(Attr_i^*, Attr_i)) + (w_n \times Sim_n(Attr_i^*, Attr_i)$$
$$| w_v \times (Sim_v(Attr_i^*, Attr_i)) + w_p \times Sim_p(Sim_p(Attr_i^*, Attr_i))$$

Where Sim_t is the similarity on data type, Sim_n is the similarity on the attribute name , Sim_v is the similarity on the attribute's list values and Sim_p is the similarity on their parents.

Algorithm1:

```
For each $Attr_i$* Do
    For each node $N_j$ and not mapped Do
        For the keyword and every synonym $Attr_{ju}$ Do
            Compute $Sim_n(Attr_i*, Attr_{ju})$
        End For
        Select Max($Sim_n$ ($Attr_i$*, $Attr_{ju}$))
        $Sim_n$ ($Attr_i$*, $Attr_j$) = Max($Sim_n$ ($Attr_i$*, $Attr_{ju}$))
        Sim($Attr_i$*,$Attr_j$)=$w_t$*$Sim_t$ ($Attr_i$*, Attrj)+ ($w_n$*$Sim_n$($Attr_i$*, $Attr_j$) | $w_v$*$Sim_v$($Attr_i$*, $Attr_j$)) + $w_p$*$Sim_p$($Attr_i$*, $Attr_j$)
    End For
    Select Max(Sim($Attr_i$*， $Attr_j$))
    If  Max (Sim($Attr_i$*, $Attr_j$)) > threshold   Then
        $Attr_i$* is mapped to $N_j$
        insert $Attr_i$* into the synonym list of $N_j$
        If $N_j$ is leave node Then
            1:1 attribute mapping
        Else m:1 attribute mapping
        End If
    Else $Attr_i$* is not mapped to any node
    End If
End For
```

The calculation of Sim_n is based on string edit distance:

$$Sim_n = 1 - \frac{dist(Attr_i^*, Attr_j)}{Max((len(Attr_i^*), len(Attr_j))}$$

The calculation of Sim_t and Sim_v are as follows:

$$Sim_t = \begin{cases} 1 & \textit{if the types are the same} \\ 0 & \textit{otherwise} \end{cases}$$

$$Sim_v = \begin{cases} 1 & \textit{if there are same values in value list} \\ 0 & \textit{otherwise} \end{cases}$$

Sim_p is calculated by the similarity of its Sim_n, Sim_t and Sim_v.

2.2.2 Data translation algorithm

(1) Simple data translation --1:1 attribute mapping

The templates in the Web database interfaces are convergent [1], there are only 25 templates. We have observed 100 Web database interfaces from TEL-8 Query Interfaces datasets of the UIUC Web Integration Repository. The datasets conclude 8 domains which are Airfares, Books, Automobiles, Hotels, Jobs, Music Records, Movies and Car Rentals. Our statistic results show that the most popular templates are [attributes, $value], [attributes, $value ∈{D}] and [attributes, $value ⊂ {D}] and the appearance frequencies of those templates are 37%, 53% and 4%, respectively. In addition, the translation usually occurs only within the certain templates which are share the same data type. So the data translation algorithm considers these two facts: the template and the data type as algorithm2.

Algorithm2:

```
$Attr_i$*and $Attr_j$ are mapping attributes
M: the node that represents $Attr_j$.value in data hierarchy
Case target template is [attributes, $value]:
    $Attr_i$ *.value= $Attr_j$. value
Case target template is [attributes, $value {D}]:
    If values of $Attr_i$ *  ∈$Attr_j$. valuelist Then
        $Attr_i$ *.value= $Attr_j$. value
    Else // select the nearest value from list values of $Attr_i$ *
        If   type="Text", Then
            resort to data hierarchy
            If values of $Attr_i$ *  ∈M.childlink.values Then
                convert the query into the union of subqueries
                in each subquery $Attr_i$ *.value = M.$child_k$.value
            End If
            If   values of $Attr_i$ *  ∈M. parent.valuelist Then
                $Attr_i$ *.value = M.parent.value
            End If
        End If
        If   type="Numeric" or type="Date Time" Then
            Select   |Min ($Attr_i$ *.value - $Attr_j$. value)|
            $Attr_i$ *.value= $Attr_j$. value
        End If
    End If
Case target template is [attributes, $value {D}]//usually the attributes are Numeric type or Data Time type
    If   $Attr_j$. value range   $Attr_i$ *.value range Then
        $Attr_i$ *.values range = $Attr_j$. values range
    Else Project   $Attr_j$. values range to number or time axis
        Select the minimal and subsume values ranges
    End If
End Case
```

(2) Complex Data translation--M:1 attribute mapping

We note that the query translation does not simply translate each constraint separately. If some constraints depend on each other, they should be translated together.

Hierarchy concept has been constructed to notify the dependency of attributes. And algorithm1 has found out not only the mapping relationship of the attributes (i.e. 1:1 attribute mapping or m:1 attribute mapping) but also the mapping type of them. We propose a type-based algorithm3 to translate attributes with different types.

Algorithm3:

```
$Attr_i$ *and $Attr_j$ are M:1 mapping relationship
case type="Text":
    Attri*.value is the string connected by a predefined order
case type="numeric":
    $Attr_i$ *.value is the sum of the n values of the attributes
case type="date time":
    $Attr_i$ *.value is composed by n values of the attributes according to a predefined format
End case
```

3. Experiment

Data set & performance measure: Our experiment is based on 20 query interfaces of different web databases collected from airfares domain. We choose airfares domain because the interface structure in this domain is more complex than that of others. These query interfaces are randomly divided into two disjoint groups. 10 query interfaces in group 1 are used for building integrated interface; the others are used for further test.

We build the integrated interface according to [10]. In the experiment, we submit queries to the integrated interface and translate them to the test interfaces. Then we consider both results of attributes mapping and data translation to measure the performance of our methods.

Experimental results: The results of evaluation are shown in Table 1 and Table 2. The column labeled as "A" gives the numbers of attributes actually extracted from query interface. The columns labeled as "R1" and "R2" give the numbers of attributes correctly matched and the final correct numbers after data translation respectively. Columns under "P1" and "P2" give their precision, respectively.

As we can see from Table 1, the user's query can be translated to local interfaces in group 1 perfectly. Although there is a certain decline in accuracy of group 2, the average precision shows that our method achieves very good results.

Table 1. The results of evaluation

	A	R1	R2	P1	P2
group 1	121	121	121	100%	100%
group 2	107	102	99	95.33%	92.52%
total	228	223	220	97.81%	96.49%

Table 2 reports the detail results of group2. The main reasons for the drop in precision are the complexity of attributes and the diversity of the naming. The errors usually happen when semantically related attributes and groups of attributes are organized in a different way. This problem is difficult to deal with and will be left for our future work.

Table 2. The results of group 2

	Interface	P1	P2
1	us.flyasiana.com	100%	100%
2	www.airfareplanet.com	100%	100%
3	www.cheapflight.com	100%	100%
4	www.cheapticket.com	100%	87.50%
5	www.continental.com	92.86%	85.71%
6	www.economytravel.com	91.67%	91.67%
7	www.flights.com	100%	100%
8	www.priceline.com	100%	100%
9	www.orbitz.com	100%	85.71%
10	www.flightcentre.us	78.57%	78.57%
	total	95.33%	92.52%

4. Conclusion

In this paper, we propose an approach to automatically translate query from the integrated interface to a Web database interface. Our algorithms can be used for not only 1:1 simple query translation but also m:1 complex one. In order to support the translation, we construct two kinds of hierarchies, one is concept hierarchy, and the other is data hierarchy. Empirical evaluation conducted shows that our approach is highly effective and efficient.

References

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

2. He B, Chang K. C.-C, Han J. Discovering Complex Matchings across Web Query Interfaces: A Correlation Mining Approach[C]// *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* Seattle, ACM Press, 2004: 148-157.

3. He B, Chang K. C.-C: Making holistic schema matching robust: an ensemble approach[C]// *The ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Chicago, ACM Press 2005:429-438.

4. He B and Chang K. C.-C. Statistical schema matching across web query interfaces[C]// *Proceedings of the ACM SIGMOD International Conference on Management of Data.* San Diego ACM Press, 2003: 217-228.

5. Wu W, Yu C, Doan A, and Meng W. An interactive clustering-based approach to integrating source query interfaces on the deep web[C]//*Proceedings of the ACM SIGMOD International Conference on Management of Data.* Paris, ACM Press, 2004: 95 -106.

6. Wang J, Wen J, Lochovsky F. Instance-based Schema Matching for Web Databases by Domain-specific Query Probing[C]// *The International Conference on Very Large Database*, Toronto. ACM Press,2004: 408-419.

7. Wu W, Yu C. WebIQ: Learning from the Web to Match Deep-Web Query Interfaces[C]//*Proceedings of the 22nd International Conference on Data Engineering.* Atlanta: IEEE Computer Society, 2006: 44.

8. Chang K. C.-C, Garc´a-Molina H. Mind Your Vocabulary: Query Mapping Across Heterogeneous Information Sources[C]// *Proceedings ACM SIGMOD International Conference on Management of Data. Philadelphia*, ACM Press, 1999: 335-346.

9. Zhang Z, He B, and Chang K. C.-C. Light-weight domain-based form assistant: Querying Web Databases on the Fly[C]// *Proceedings of the 31st International Conference on Very Large Data Bases.* Trondheim, ACM Press, 2005: 97-108.

10. Dragut E, Wu W, Sistla P. Merging Source Query Interfaces on Web Databases[C]//*Proceedings of the 22nd International Conference on Data Engineering.* Atlanta: IEEE Computer Society, 2006: 679-690