Research on Web Query Translation based on Ontology

Xin Wang
Changchun Institute of Technology/College of Software, Changchun, China
Email: wangxccs@126.com

Ying Wang
Jilin University/College of Computer Science and Technology, Changchun, China
Email: {wangying2010}@jlu.edu.cn

***Abstract*—This paper presents a framework of query translation based on ontology between different query interfaces (forms), which will translate the queries into suitable formats, hiding the differences from underlying sources, and enabling users to access data sources conveniently. Firstly, constructing domain ontology concept model (DOCM), then, finding out predicate matching relationships between source form and target forms based on DOCM, lastly, analyzing query rewrite strategy which contains four type rewriters to construct automatically a set of constraint mapping rules so that the system can use the user-provided input values to fill out the appropriate form fields of different web database forms. Experiment results show that this method is feasible and effective.**

***Index Terms*—Ontology, Query Translation, Predicate Matching, Query Rewrite**

I. INTRODUCTION

Large portions of web are buried behind user-oriented interfaces (forms), which can only be accessed by filling out forms. When a user poses all the queries on a global query form, web query translation system will rewrite the queries into suitable formats for the sources, and automatically submit the queries to different underlying physical sources, making the accesses to individual physical sources transparent to the user[1]. Due to integrating multiple heterogeneous data sources to provide users with a unified view, web query translation will be a complex and challenged key point.

One of the earliest efforts at automated form filling was the ShopBot project[2], which uses domain-specific heuristics to fill out forms for the purpose of comparison shopping. The ShopBot project, however, did not propose a general-purpose mechanism of filling out forms for non-shopping domains. Raghavan,S. and Garcia-Molina,H.[3] build Hidden Web Exposer (HiWE), it does this by filling out the searchable form, submitting one or more queries, and after the contents have been extracted, it stores them in a repository and builds an index to support user queries. Shu L et al.[4] propose a method to model the querying capability of a query interface based on the concept of atomic queries. Meanwhile, they also present an approach to construct querying capability automatically. Fangjiao Jiang et al.[5] study the query translation problem of the exclusive query interface and present a novel Zipf-based selectivity estimation approach for infinite-value attribute. Experimental results on several large-scale Web databases indicate that the approach can achieve high precision on selectivity estimation of infinite-value attribute for exclusive query translation. Yongquan Dong et al.[6] propose a novel query interface matching approach based on extended evidence theory for Deep Web. Against the limitations of existing combination methods, their approach considers the credibilities of different matchers and incorporates them into exponentially weighted evidence theory to combine the results of multiple matchers. Tan et al. [7] introduce personalization recommendation to the Deep Web data query, proposed a user interest model based on fine-grained management of structured data and a similarity matching algorithm based on attribute eigenvector in allusion to personalization recommend-ation.

A few efforts are dedicated to query translation which has responsibility for translating a user's query from source form to target forms. However, research works related to query translation have mainly fallen into two categories: attribute mapping and constraint mapping. These methods do not consider applying background knowledge, which is important to understand problems and situations. Ontology is an explicit, formal specification of a shared conceptualization in a interesting domain, therefore, we propose a novel method of ontology-assisted query translation between different query forms in this paper.

II. WEB QUERY TRANSLATION ANALYSIS

Web query translation problems can be defined as translating a user's query from a source form to different target forms (Fig.1), which contains three characteristics:

doi:10.4304/jsw.7.12.2849-2856

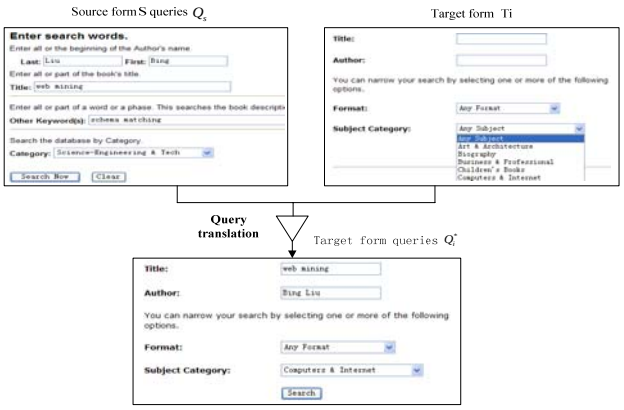

Figure 1. The query translation between source form and target form, where denotes the source form, and denotes the target form.

Characteristic1 Query-generally: the queries of target forms are similar with the queries of the source form.

Characteristic2 Domain-generally: the source form and the target forms belong to a domain-specific.

Characteristic3 Minimum subsumption-generally: web query translation does not miss any correct answer and contains fewest incorrect answers. Simultaneity, it is domain content independent.

Definition1 Query Translation: Given a source form S and a set of target forms T = { T_1, T_2, …, T_n}, if the user-provided input value is Q_s in the source form S, then, the output is the query combination Q_t^* = {Q_1^*, Q_2*, …, Q_n*} in the target forms after query translator, $Q_i^* = \delta_i(q_1, q_2, ..., q_n)$, $1 \le i \le n$, which satisfies the following constraints[8][9]:

(1) Each query q_j $(1 \le j \le n)$ in Q_i^* must be a valid query to target form T_i .

(2) Q_i^* is "close" with the source queries Q_s as much as possible in order to minimize the processing costs of target form, and improve the retrieval quality.

Definition2 Minimum Subsumption Strategy (*MinSS*): Given a global query Q_s from source form Q_s and a set of target forms T = { T_1, T_2, …, T_n}, $Q_i^*(1 \le i \le n)$ is a web query translation using *MinSS* if the following three conditions are satisfied at the same time:

(1) Q_i^* is a valid query of target form, namely, Q_i^* is acceptable to target form T_i .

(2) Q_i^* semantically subsumes Q_s, namely, for any back-end database $D, Q_s(D) \subseteq Q_i^*(D)$, it means that the retrieved information of Q_s is a subset of Q_i^*.

(3) Q_i^* is the minimum subsumption strategy, that is to say, there does not exist any query Q_t which satisfies (1) and (2) and $Q_t(D) \subseteq Q_i^*(D)$.

MinSS can minimize the amount of data to be sent from the web database and minimize the effort to filter out undesired results.

III. Query Translation Framework

In order to realize query translation between different query forms, we need to reconcile the heterogeneities at the predicate-level and value-level, and then generate a query plan expressed upon each target form. The framework of web query translation based on ontology is shown in Fig.2.

Predicate recognition module is used to find out the attribute(predicate) matching relationships between the source form and the target forms based on ontology; Predicate mapping is used to match the source form and the target forms to determine the mapping type; Query rewrite module is used to match the value translation of mapped predicate; Query submission module is used to trigger button to submit query plans with respective sources, so that the user can immediately receive query results from multiple data sources.

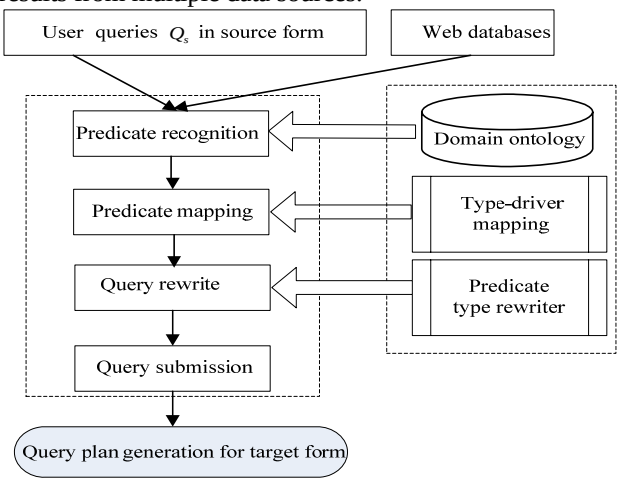

Figure 2. The framework of web query translation, which mainly contains predicate recognition, predicate mapping, query rewrite and query submission.

A.Ontology Construction

Ontology as the concept model describing information system in semantic and knowledge level, user's queries and relevant data can be mapped to the concept model, thus, ontology can be seen as a knowledge system which describes concepts and relationships[10][11]. The process of ontology construction is shown in Fig.3:

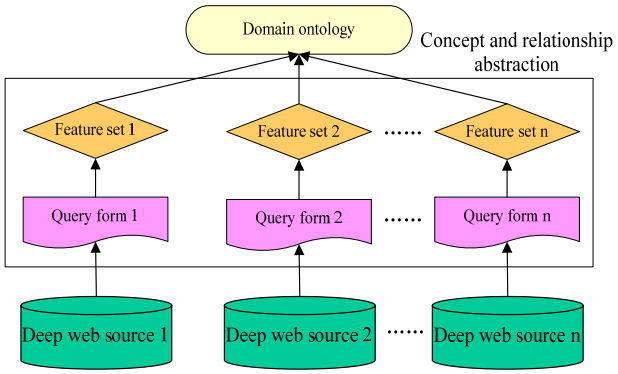

Figure 3. Ontology construction: the process is semi-automatic, the concepts and relationships of ontology are extracted from searchable forms.

The process of ontology construction is as follows:

(1) Extracting domain terminologies from searchable forms, simultaneously, capturing the semantic relationships of terminologies.

(2) Extracting domain concepts. The domain concepts need explicit, so there is a need to disambiguate terminology, and describe the terminology with the most common word.

(3) Constructing the hierarchical relationships of concepts, such as Is-A, Part-Of, Synonymous etc.

(4) Refining domain ontology. Ontology engineers revise concepts and semantic relationships under the guidance of ontology engineering standards to ensure that the domain knowledge is shared and consistent.

(5) After constructing the ontology, the semantics of concepts in domain ontology can be enriched by adding concept-level knowledge based on three philosophical notions: identity, rigidity, and dependency.

Definition 3. Domain Ontology Concept Model (DOCM): DOCM is a data model that describes a set of concepts and relationships that may appear in a specific domain. It should be understandable by machine so that it can be used to reason about these objects within that domain. Each object can be denoted as *Class* = {*CM*, *DT*, {S_i}, {CA_i}, {SC_i}}, which describes the relevant information of object.

CM: It denotes the main class of object, which is universal and easy to understand for users. It can be seen as the keyword of object, e.g. "Author" can be seen *CM* of name object in Book domain.

DT: It denotes the data type of object, such as "string", "numerical" and so on, e.g. the *DT* of "Author" is "string".

{S_i}: It denotes the synonyms of *CM*, namely, the concept aliases, e.g. "Writer" is the aliases of "Author".

{CA_i}: It denotes the condition property set of object, which is "Part-Of" relationship to *CM*, e.g. "First name" and "Last name" are two condition properties of "Author".

{SC_i}: It denotes the sub class set of *CM*, which is "Is-A" relationship to *CM*. Such as "Author keywords" and "Title keywords" are "Is-A" relationships to *CM* "Keywords".

DOCM has a good organizational structure, which represents high-level background knowledge with concepts and relationships. In this paper, the ontology is implemented by Protégé API and represented in the Web Ontology Language(OWL)[12]. In this way, to operate DOCM is equivalent to operate the OWL file.

B. Predicate Recognition

Predicate recognition is used to find out the predicate matching relationships among different query forms, an example is shown in Fig.4.

Figure 4. Predicate recognition of three query forms in Book domain, we can find predicate correspondences such as those indicated by different shapes or underlined.

The predicate matching relationships between the source form and the target forms are complex as follows:

1) The predicate labels belong to the same concept are always different, e.g. source form denotes name using "Author", while target form using "Writer".

2) The complex predicate relationships are common, e.g. source form denotes time using "data", while target form using "year, month, day".

Predicate recognition can be seen as schema matching for different query forms, which are classified attribute-level (predicate-level) matching(Fig.5).

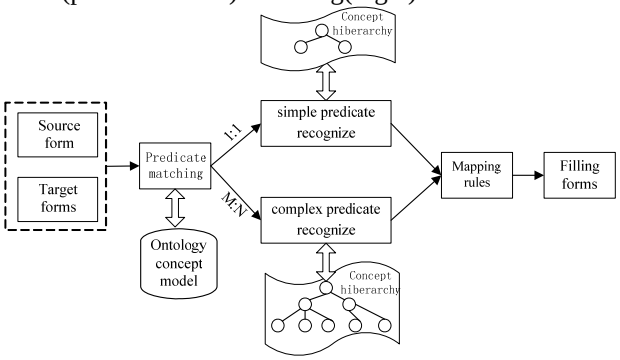

Figure 5. Predicate recognition, which can be divided into simple predicate recognition(SPR) and complex predicate recognition(CPR).

Definition4 Simple Predicate Recognition (SPR): the predicate of source form and the predicate of target form is 1:1 matching.

Definition5 Complex Predicate Recognition (CPR): the predicate of source form and the predicate of target form is M:N matching.

Definition6 Ontology-assisted Predicate Recognition Algorithm: the semantic similarity between two predicate concepts is measured by DOCM, which is the "bridging" to obtain the predicate matching relationships among different query forms. Assume that A^* is a predicate label from target form, A_i is the main class of concept node C_i from DOCM, {S_i} is the synonym set of A_i, {CA_i}is the condition property set of concept node C_i, {SC_i} is the sub class set of concept node C_i, $Sim(A^*, A_i)$ denotes the similarity between A^* and A_i, σ denotes the threshold of matching degree:

a) If $A^* \in \{S_i\}$ or $A^* = A_i$, then it is 1:1 matching(SPR) between predicate label A^* of target form and ontology concept A_i.

b) If $A^* \notin \{S_i\}$, $A^* \neq A_i$, and $Sim(A^*, A_i) \geq \sigma$, then it means 1:1 matching (SPR) between predicate label A^* and ontology concept A_i, we add A^* to {S_i} as a new synonym of ontology concept A_i.

c) If $A^* \in \{CA_i\} \cup \{SC_i\}$, then it is M:1 matching (CPR) between predicate label A^* of target form and ontology concept A_i.

d) If $A^* \notin DOCM$ and $\forall A_i$, $Sim(A^*, A_i) < \sigma$, then it means there is no matching between predicate label A^* of target form and ontology concept A_i, we add A^* to DOCM as a new ontology concept.

e) For the other target forms, we do the same above, in this way, we can get ontology-form predicate matching table, which records the matching relationships between different forms and ontology concepts. The structure of predicate matching table is shown in Fig.6.

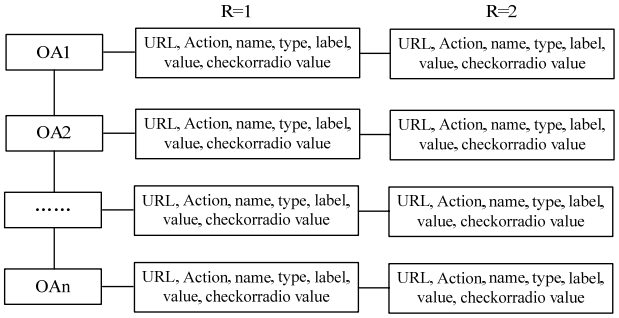

Figure 6. Predicate recognition, which can be divided into simple predicate recognition(SPR) and complex predicate recognition(CPR).

We exploit the "bridging" effect in the process of matching predicates based on DOCM from a large number of query forms. Through predicate recognition, we can capture the semantic relationships of predicates among different query forms.

C. Predicate Mapping

Predicate mappings are crucial for the system to reformulate a user query into queries on the sources. If we can understand their semantics, we can query in the target forms to find the nearest queries for query optimization. Due to the predicate types are not single from different forms, relying on a single predicate type is not sufficient and the match results of individual types are often inaccurate and uncertain. Literature [13] finds that the most commonly used predicate templates are [*attribute*; *value*] and [$attribute; value \in \{D\}$] by observing 150 web query forms. Through observation, we found that the predicate type mainly contains "text", "select", "numeric", "data time" and "specific type". Therefore, we propose a type-based approach for query rewrite by combining multiple type rewriters to obtain the appropriate matching value.

Predicate mapping is used to match source form and target forms to determine the mapping type. The structure of predicate mapping is shown in Fig.7.

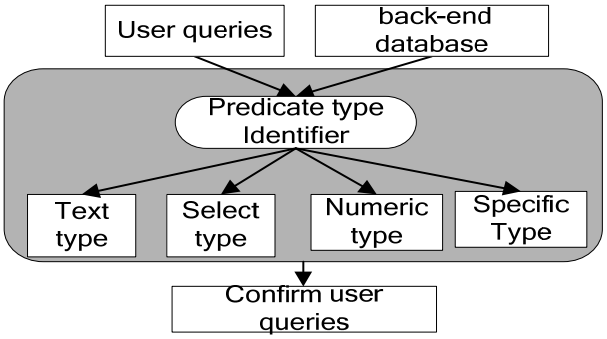

Figure 7. The structure of predicate mapping, which contains text type, select type, numeric type and specific type.

We define a predicate template as a four-tuple [*attribute*; *type*; *value*; *constraint*] in this paper, where *attribute* denotes the predicate label, *type* denotes the predicate type, *value* denotes the filling value of predicate, and *constraint* denotes the predicate constraint condition. For example, in Fig.4(b), if a user fills out "Deep Web" in the predicate "Title/Keywords" and selects "All Terms" as the constraint condition, then, its predicate template can be expressed with [*Title/Keywords*; *text*; *DeepWeb*; *AllTerms*]. For the target forms, according to different predicate type, predicate template can be divided into text predicate template [*attribute*; *text*; *value*; *constraint*], select predicate template[*attribute*; *select*; *value*; *constraint*], numeric and data time predicate template [*attribute*; *numeric*; *value*; *constraint*] and specific predicate template [*attribute*; *specific*; *value*; *constraint*]. When a user fills out query form, it will form the source binding template, and then recognize the corresponding type by predicate mapping, lastly, send to different type rewriters to rewrite queries.

D.Query Rewrite

Query rewrite is responsible for rewriting the query into an equivalent from that one can execute against the physical databases, which is classified value-level matching. Nowadays, query rewrite based mostly on inefficient keyword matching techniques becomes a bottleneck of the web, therefore, we need to construct automatically a set of constraint mapping rules so that the system can use the user-provided input values to fill out the appropriate form fields of different web database forms[14]. In order to guarantee *MinSS* , a query of target form field can be seen as a union query Q_t^* which is a union of queries upon the target form to relevant answers from the target database. We want the union query Q_t^* to be as "close" to the source query Q_s as possible so that it retrieves fewest extra answers.

Generally, these requirements of form filling are commonly referred to as access limitations in that access to data can only take place according to given patterns, namely, search space[15].

Definition7 Search Space. For a predicate *P*, we define search space of *P* is *I(P)*, which denotes all possible instances of *P*.

Query rewrite based on type can be seen as a search problem, its search space can be confirmed by the predicate type. Each type of rewriter implements a search for the type-driven mechanism, if the predicate value is W_s for source form, then its value is W_s for the absence of a predefined target value. If there exists a predefined search space in target form, its value can be obtained from predefined values, namely, can be obtained from search space[16]. Each type rewriter has a quite different performance in rewriting queries. To obtain the final matches, we use some heuristics to perform rewriting strategy.

(1) Text type rewriter: If the predicate type is "text" in target form, namely, its predicate template is [*attribute*; *text*; *value*; *constraint*], then filling out this text control of target form with the same value of source form, simultaneity, filling out this constraint value with the

same value of source form. If there does not exist constraint condition for the predicate in target form, then *constraint* is set *null* . For select type rewriter, numeric type rewriter and specific type rewriter, we adopt the same approach to obtain the *constraint* value of target form.

(2) Select type rewriter: If the predicate type is "select" in target form, namely, its predicate template is , and its search space is the string instances of select list, then, we need to compute the matching degree between the source predicate value and the target search space, lastly, selecting the value by using null correspondences strategy[17] as the union query (filling value) of target form, in which we no longer need a threshold to filter out likely incorrect value correspondences. In this union query, the one semantically closest to the source predicate value can be seen as the best translation.

Definition8 Null Correspondence: Given a string *s* and a set of string $\{t_i\}$(i=1, 2, …, n), null correspondence of the string *s* based on the set of string $\{t_i\}$ is defined as formula (1):

$$simn(s, null) = \prod_{i=1}^{n}(1 - simp(s, t_i)) \quad (1)$$

Where *simp*(*s*, t_i) denotes the semantic similarity between the string *s* and the string t_i. When we get the value of null correspondence, *simn*(*s*, *null*) will compare with all *simp*(*s*, t_i) (i=1, 2, …, n). If *simp*(*s*, t_i) is the maximum value, then it means the string *s* is mostly similar with the string t_i, so t_i will be selected as the filling value; If the maximum value is *simn*(*s*, *null*), then it means the string *s* is not similar with all string t_i (i=1, 2, …, n), so null or default value will be filled into the query field.

There are two cases for "select" types:

1) If the predicate type is "select" in source form, namely, its predicate template is [*attribute*; *select*; *value*; *constraint*], the predicate type in target form is also "select". Since the set of select values is fixed, there is no possibility of mismatched for users. In this case, we can use the ontology-assisted similarity algorithm to obtain the desired union query.

2) If the predicate type is "text" in source form, namely, its predicate template is [*attribute*; *text*; *value*; *constraint*], the predicate type in target form is "select". While not all the users are professional, they may not know how to exactly describe what they want and sometimes they type a wrong word into the query field. Therefore, query system may possibly return some results the user doesn't want or even no result at all. Such a query is called failed query and this will bring on some uncertainties. In order to obtain a fixed quantity of results satisfying the query to the user, a query strategy is proposed: firstly, computing similarity based on edit distance matcher which is explained in definition 10, if the similarity does not exceed a threshold, then it means the two phrases are not shaped in font, in this case, switching to call ontology-assisted similarity algorithm.

Definition9 Edit Distance Matcher: Given two character strings S_1 and S_2, edit distance between S_1 and S_2 is measured by the number of edit operations, including insertions, deletions and substitutions, necessary to transform one string into another. We define the similarity based on edit distance between two strings (phrases) S_1 and S_2 as formula (2):

$$Sim(s_1, s_2) = \frac{1}{1 + ed(s_1, s_2)} \quad (2)$$

Where *ed* (S_1, S_2) is the edit distance of string S_1 and S_2.

(3) Numeric type rewriter: the predicate type is "numeric" in target form, namely, its predicate template is [*attribute*; *numeric*; *value*; *constraint*]. Numeric and data time have very similar nature in that they all form a linear space, and have similar operators, therefore, the mapping techniques for the two types are generally the same.

For the choice of numeric or data time can be divided into two ways, one is the discrete data selection, the other one is the continuous data selection. For the discrete data selection method, we call discrete numeric data matcher which is explained in definition11.

Definition9 Discrete Numeric Data Matcher: If the user-provided input value in source form is *q*, a discrete data. The corresponding search space in target form is $D = \{q_1, q_2, \ldots, q_n\}$, where, $q_1, q_2, \ldots, q_n$ are discrete data, then the similarity between q and q_i ($1 \le i \le n$) can be defined as formula (3):

$$Sim(q, q_i) = 1 - \frac{|q - q_i|}{\max(q, q_i)} \quad (3)$$

Assume σ is the threshold of MD, if $Sim(q, q_m) = Max\{Sim(q, q_1), Sim(q, q_2) \ldots Sim(q, q_n)\} \ge \sigma$, $1 \le m \le n$, then we select q_m as the union query of target form.

Example1 If the user-provided input value in source form is "100" for predicate label "Price", the corresponding search space in target form is {90,95,98}, $\delta = 0.8$, then through the similarity formula (2), we can infer that *Sim*(100,90) = 0.9, *Sim*(100,95) = 0.95, *Sim*(100,98) = 0.98, thus, the value "98" of search space is the most greatest similarity with "100", thus, we should select "98" as the union query of target form.

In query forms, the user is likely to be given ranges of values to choose(Fig.8), to help the query rewriter understand different ranges, we need to let the system know the meanings of the range modifiers. For this purpose, we build a semantic dictionary that keeps commonly used range modifiers for numeric domains (Table1).

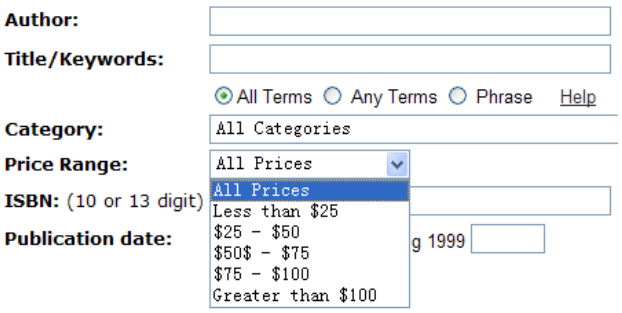

Figure 8. An example for rang modifiers, we can see that the ranges are often formed using numeric values and range modifiers together.

TABLE I.
THE SEMANTIC DICTIONARY FOR RANGE MODIFIERS, AND WITH THE INFORMATION AND AN APPROPRIATE SEMANTIC DICTIONARY FOR RANGE MODIFIERS, WE CAN INFER THE CORRESPONDING NUMERIC RANGES FROM RANGE SEMANTICS TABLE.

Range modifiers	Meaning
More than	>
Less than	<
under	<
from	>=
……..	……..
Any	All range

After converting the range modifiers, we need to select the advisable value as the filling value. In order to reduce the undesired intermediate results to the minimum, we ought to choose the minimum target range, namely, the target range is exactly a minimum subsumption of the original one. The most appropriate value can be easily found out by projecting the original value to the number or time axis, which is explained in definition12.

Definition10 Continuous Numeric Data Matcher: If the user-provided input value in source form is q, which is a continuous data with range [a,b], the corresponding target range is q_i, whose selective values are continuous ranges S, $S=\{[n_1,n_1^t],[n_2,n_2^t],...[n_m,n_m^t]\}$, $n_i \le n_i^t$, $n_i^t \le n_{i+1}$, $1 \le i \le m-1$, the overlap grade between q and S reflects the MD. There are four cases for the overlap grade between $[a, b]$and $[n_i,n_i^t]$:

a) $n_i \le a$ and $a \le n_i^t \le b$

b) $a \le n_i$ and $n_i^t \le b$

c) $a \le n_i \le b$ and $n_i^t \ge b$

d) $n_i \le a$ and $n_i^t \ge b$

In this condition, the union query is the conjunction of the above four cases, namely:

$$q_i = Condition_a \cup Condition_b \cup Condition_c \cup Condition_b$$

If the source query range becomes large, the cost of *MinSS* will be high because the number of targets queries to the web database will increase. An example for minimum subsumption strategy is shown in Fig.9.

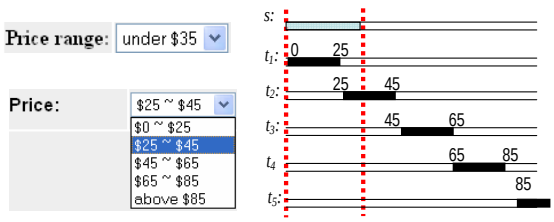

Figure 9. An example for minimum subsumption strategy: For the predicate label "Price range"(Fig.9), if the filling value is "under 35" in source form, whose meaning is the value of less than 35 by range modifiers table. The search space in target form contains five selective values "0-25", "25-45", "45-65", "65-85", "85-", we can observe that the projection area in source form contains two target search space "0-25" and "25-45" by numeric projection, that is to say the minimum subsumption using ^{MinSS} in target form is {"0-25", "25-45"}, namely, the conjunction of target form which contains the source projection area.

(4) Specific type rewriter: The above type rewriters can only deal with 1:1 mapping to find corresponding filling values, which can not handle complex mapping. Complex mapping binds a source schema element to a target schema element through an appropriate mapping expression. Therefore, we need specific type rewriter to realize the complex mapping by DOCM, which the user-provided input values will be stored to corresponding class. If the predicate type is "text", then, according to a predefined order, specific type rewriter will assign the value of source form to target form; if the predicate type is "numeric", then, according to a predefined format, specific type rewriter will assign the value of source form to corresponding field of target form.

E. Query Submission

After query rewrite, user's suggested queries have been filled out each target forms, then the system will automatically trigger button to submit these query plans. As we know, when a form is submitted, the web browser will code the data which the user fills out source form, the final query forms for sending web server are in accordance with the "variable name 1 = value & variable name 2 = value & ...", the data format as URL parameters affixing to this basic URL address to realize the complete URL address for web server. To sum up, we can see that to complete the automatic submission of each target form, it needs to achieve this function of simulating browser, namely, constructing URL address with parameters and triggering this URL. In this case, the query posted to the source form can be precisely translated to a query against the target form.

Example2 Given a URL http://www.abbeys.com.au/search.asp, we can observe that the base URL is http://www.abbeys.com.au, and the action path is /search.asp, the form fields are "Author", "Keywords", "Category", "Price Range", "ISBN" and "Publication data", if we fill out this form with value "Bing Liu"、"web mining"、"Computing and information technology" 、 "All Prices"、"After the year 2000" respectively, then, the query is thus constructed as: http://www.abbeys. com. au /search.asp？Author=Bing+Liu&Title/Keywords =web+ mining&Category=Computing+and+information+technol ogy&PriceRange=All+Prices&ISBN=&Publication data= After+the+year+2000. If we send this query directly to the web site, we will get the desired results from the target site[18].

IV. EXPERIMENTS

Though the above analysis, we implement the graphical interface for web query translation which is shown in Fig.10.

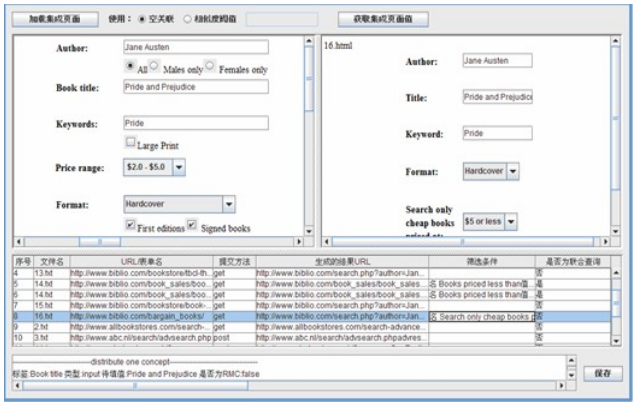

Figure 10. The graphical interface for web query translation.

To evaluate our approach, we select the Book-Domain query interfaces from UIUC[19] as the test samples. The evaluation metric for query translation is measured by Precision, Recall and F-measure. Precision denotes the percentage of the correctly query translation templates over all the query translation templates, Recall denotes the percentage of the correctly query translation templates over all the query form, and F-measure is a comprehensive assessment of Precision and Recall. The formulas are defined as (4), (5) and (6):

$$\text{Precision} = \frac{\text{the correctlyquery translation templates}}{\text{all thequery translation templates}} \times 100\% \quad (4)$$

$$\text{Recall} = \frac{\text{the correctly query translation templates}}{\text{all the query form}} \times 100\% \quad (5)$$

$$F\text{-}measure = \frac{(1+\mu^2) \times Precision \times Recall}{\mu^2 \times Precision + Recall} \times 100\% \quad (6)$$

Where μ denotes the weight coefficient, which is used to measure the importance between Recall and Precision.

The results are shown in Table.2:

TABLE III.
THE RESULTS OF QUERY TRANSLATION FROM BOOK-DOMAIN QUERY INTERFACES.

Form number	Predicate template number	Predicate template Number correctly	Recall	Precision	F-measure
25	132	123	0.884	0.932	0.907
50	256	235	0.869	0.922	0.895
75	364	337	0.867	0.926	0.896
average			0.873	0.927	0.899

From the result of Table2, we can see that the query translation accuracy is stabilized and high with the increase of form numbers, and it indicates that the query translation method with ontology is feasible and effective.

In order to verify the validity of various predicate types during the process of query translation, we measure it by the precision of predicate mapping, the results are shown in Table.3:

TABLE II.
THE MAPPING RESULTS OF FOUR DIFFERENT PREDICATE TYPES

Predicate type	Predicate template number	Predicate template Number correctly	Precision
Text	135	129	95.6%
Select	73	66	90.4%
Numeric	17	15	88.2%
Specific	26	21	80.8%
Total	251	231	92.0%

From the results of Table3, we can see that the predicate type "text" occupies the largest number in all predicate types, and the precision is also the best. The main reason for "text" type translation mistake is the predicate matching error, and the text field does not rationally combine with corresponding constraints in the process of schema extracting. The mistake reason for "select" type is to mistakenly input dissimilarity value and not to fill in the real similarity value in computing similarity based on edit distance. The inconsistent representation for "numeric" type content is the main reason to cause mistake. The "specific" type is more complicated, the mistake is mainly due to unreasonably split or combine source query, partly to schema extraction and predicate matching. The statistical results show that the precision of four types predicates mapping have reached a higher level, but the precision of "specific" type query translation would be further improved.

V. CONCLUSION

We presented a novel approach to query translation based on ontology automatically, which can transform queries from source form into queries that address the underlying physical databases. The experiment results show that our approach is feasible and effective.

For our future work, we plan to enrich the domain ontology and continually improve query translation algorithm, we will also further research on the complex matching problem. We believe that our approach, with appropriate extensions, can achieve better results.

REFERENCES

[1] Fan Wang, Gagan Agrawal, and Ruoming Jin. Query Planning for Searching Inter-dependent Deep-Web Databases. In Proceedings of SSDBM, 2008, pp24-41.
[2] R.B.Do orenbos, O.E tzioni, and D.S.W eld. A scalable comparison-shopping agent for the world wide web. In Proceedings of the First International Confence on Autonomous Agents Marina del Rey, 1997, pp39-48.
[3] Raghavan,S. and Garcia-Molina,H. Crawling the hidden web. In Proceedings of the 27th International Conference on Very Large Data Bases (VLDB), 2000, pp129-138.
[4] Shu L, Meng W, He H, et al. Querying capability modeling and construction. In Proceedings of the 8th International Conference on Web Information Systems Engineering (WISE), 2007, pp13-25.
[5] Fangjiao Jiang, Weiyi Meng, Xiaofeng Meng. Selectivity estimation for exclusive query translation in deep web data

integration. Lecture Notes in Computer Science(LNCS) Journal, 2009, 5463: 595-600.
[6] Yongquan Dong, Qingzhong Li, Yanhui Ding, Zhaohui Peng. A query interface matching approach based on extended evidence theory for deep web. Journal of Computer Science and Technology, 2010, 25(3): 537-547.
[7] Tao Tan, Hongjun Chen. A Personalization Recommend-ation Method Based on Deep Web Data Query. Journal of Computers, 2012, 7(7): 1599-1606.
[8] K.C.-C.Chang and H.Garcia-Molina. Approximate query mapping: accounting for translation closeness. VLDB Journal, 2001.
[9] K.C.-C.Chang and H.Garcia-Molina. Approximate query translation across heterogeneous information sources. In Proceedings of the 26th VLDB Conference, 2000, pp566-577.
[10] Kerui Chen, Wanli Zuo, Fan Zhang, Fengling He, Yongheng Chen. Robust and Efficient Annotation based on Ontology Evolution for Deep Web Data. Journal of Computers, 2011, 6(10): 2029-2036.
[11] Weifeng Su, Jiying Wang, Frederick H.Lochovsky. ODE: Ontology-assisted data extraction. ACM Transactions on Database Systems, 2009, 34(2):1-35.
[12] Matthew Horridge, Bijan Parsia, Ulrike Sattler. Explanation of OWL entailments in Protege4. In proceedings of International Semantic Web Conference, 2008.
[13] Zhen Zhang, Bin He, Kevin Chen-Chuan Chang. Light-weight domain-based form assistant: querying web databases on the fly. In Proceedings of the 31st Very Large Data Bases Conference (VLDB), 2005, pp97-108.
[14] Jie Bao, Doina Caragea, Vasant Honavar. Query translation for ontology-extended data sources. American Association for Artificial Intelligence, 2007.
[15] Andrea Cali, Davide Martinenghi. Querying the deep web. EDBT, 2010.
[16] Ye Ma, Derong Shen, Yue Kou, and Wei Liu. An effective query relaxation solution for the deep web. In Proceedings of APWeb, 2008, pp649-659.
[17] Zhongtian He, Hong Jun, D A.Bell. A Prioritized Collective Selection Strategy for Schema Matching across Query Interfaces [C]. Proceedings of the 26th British national conference on Databases (BNCOD'09), LNCS 5588, 2009: 21-32.
[18] Stephen W.Liddle,David W.Embley, Del T. Scott. Extracting data behind web forms. Proceedings of the 28th VLDB Conference, 2008, pp402-413.
[19] The UIUC Web Integration Repository. http://metaquerier.cs.uiuc.edu/repository.

Xin Wang was born in HuLuDao, LiaoNing Province, China, on October 21, 1981. He received his Master of computer science from Harbin Engineering University in 2008. Currently, he is a PH. D candidate with computer science at Jilin University since 2011. His main research interests include information retrieval, machine learning, and web mining.

Ying Wang was born in 1981. She is a lecturer at the Jilin University and a CCF member. She received her Ph.D. degree from Jilin University. Her research area is Web Information Mining, Ontology and Web search engine.