Understanding Query Interfaces: Automatic Extraction of Data from Domain-specific Deep Web based on Ontology

Li Dong, Zhang Huan and Yu Zitong
China Electronic Product Reliability and Environmental Testing Research Institute, Dongguanzhuang Road No.110, Guangdong province, China

https://www.ceprei.com/

Keywords: Deep Web Mining, Domain Ontology, Schema Extraction, Query Transformation.

Abstract: The resources of many Web-accessible databases, which are a very large portion of the structured data on the Web, are only available through query interfaces but are invisible to the traditional search engines. Many methods, which discovery these resources automatically, rely on the different structures of Web pages and various designing modes of databases. However, some semantic meanings and relations are ignored. Here we introduce a Web information retrieval system that obtains the knowledge from multiple databases automatically by using common ontology *WordNet*. Also, deep Web query results are post-processed based on domain ontology. That is, given an integrated interface, after inputting a query, our system offers an ordered list of data records to users. We have conducted an extensive experimental evaluation of the Web information retrieval system over real documents. Also, we test our system with hundreds of databases on different topics. Experiments show that our system has low cost and achieves high discovering accuracy across multiple databases.

1 INTRODUCTION

As the Web resources from the complex WWW continues to grow at a high speed, the problem of efficient and accurate mining from such an environment also keeps growing. One especially importance facet of this problem is the ability to not only locate and access data on the Web which is hidden from typical search techniques, but also effectively integrate a unified query interface which accepts the queries from one local form, and fill the queries back to the local forms from multiple databases. Indeed, many online network databases, which have a larger amount of data, are hidden behind the query interface in Web pages. Instead of specifying a URL to send an HTTP request to get the static page information, accessing hidden Web (deep Web) resources need to submit queries to the query interface provided by the website. The query interface is the entrance to get the Web database information. Even some resources in hidden Web sites have some links that can be indexed by a traditional search engine, they may have much more information accessible only through the query interface, as the following example illustrates.

Example 1. In the past decade, a database Science Citation Index Expand ® from Institute for Scientific Information (ISI) makes article information with high-quality about 8300 major journals across 150 disciplines accessible through the Web. If we query the database for articles with “deep Web”, then the database returns 1590 matches. The articles are stored locally at the database rather than distributing over the Web. That is, the high-quality articles of SCI can not be reached by traditional search engines. Even though some articles can be indexed by static links, it is because there are navigation bars linking to the articles whose resource providers are not SCI as well.

In this paper, we concentrate on accessing to a variety of databases and discovering valuable information automatically. More specifically, for our purpose lots of information on a specific domain across different databases is achievable through source queries and a unified integrated interface. Because of the heterogeneity and autonomy of each database in deep Web (Shestakov, Bhowmick, and Lim, 2005), completely accurate query transformation cannot be achieved. So, how to guarantee the similarity between the returning queries in local forms and the source queries to the greatest

Dong, L., Huan, Z. and Zitong, Y.
Understanding Query Interfaces: Automatic Extraction of Data from Domain-specific Deep Web based on Ontology.
DOI: 10.5220/0009514202410248
In *Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS 2020) - Volume 1*, pages 241-248
ISBN: 978-989-758-423-7

extent is a challenging work. Many researches on automatic deep Web discovery have been done (Madhavan et al. 2007; Hong, He, and Bell 2009; He, Hong, and Bell 2009; An et al. 2008). He et al (2003; 2005; 2007) presented an interface layout expression called *IEXP* according to the location layout of labels and query control elements in query interface, and developed an extraction tool called *WISE-IExtractor* to get query interface schema. Liu et al. (2010) proposed an approach which primarily utilizes the visual features on the deep Web pages to implement deep Web data extraction, including data record extraction and data item extraction. He et al. (2004) built a statistical model which adopted a positive correlation and a negative correlation mining algorithm finding out the hidden complex matching patterns to analyze the frequency and pattern of attribute names occurring at the same time. Fan et al. (2011) proposed a query transformation based on predicate template. Our contributions are summarized as follows:

• Domain ontology is built under the guidance of domain specialists. In order to guarantee the correctness of ontology, we built the ontology manually. Besides, in the process of using ontology, it is updated and improved automatically and continuously.

• The research on schema extraction from deep Web query interface is a key step in hidden Web resources mining. A novel approach to extract interface schema from deep Web based on domain ontology is presented, in which a new presentation of query interface attribute is proposed.

• In addition, it is hard to avoid submitting repeated attribute information in these queries which wastes lots of time and energy. Therefore, in order to visit multiple databases in the same time and get useful information through comparing and screening returning results, it is of great significance to match multiple attributes in different interfaces and integrate a unified query interface from view of efficiency. Most traditional integrated methods of deep Web interface obtain mapping relations based on abundant statistics. They not only need lots of sample spaces, but also ignore some semantic relations between attributes. A two-phase pattern matching approach based on ontology is defined to achieve matching between form attributes and knowledge in ontology. The matching results are used to update the ontology and simplify the following matching process. An integrated query interface is generated according to the obtained mapping relations and schema information.

• Query transformation, where the query requests are submitted to multiple destination query interfaces selected by a specific domain, is a critical component of deep Web integrated system. This paper presented a type-driven minimum superset query transformation based on ontology. The predicate templates with constraints and four types of predicate processors are proposed in the query transformation, in which automatically transforming one query in the integrated form to the multiple queries in the destination query forms is achieved.

• Due to the different styles of data records extracted from multiple databases, post-processing is an important part to organize, simplify, and convert these kinds of data records into a unified structure. In the part of building domain ontology, BIM and RSEM are adopted to obtain the feature vector of domain books. In the part of extracting information, Web pages are first parsed using HTML parser. Then HTML tree is obtained after getting rid of information that users do not interested in, such as ads or navigation and so on. Finally, the relevant data blocks in HTML tree are extracted, ranked and stored.

2 EXTRACTION OF INTERFACE SCHEMA

An imperative step of discovering information in deep Web is interface schema extraction, which finds out the common characteristics in query interfaces from two perspectives, the internal code and visual unit.

Based on the controls in the query interface, texts associated with controls, and the characteristics of layout, the tags which locate in <form> and </form>, are divided into three kinds. The definitions of tags are as follows:

Definition 1. ***Control Tag*** **(*CT*)** *is used to receive the query value which is input from users and mainly described with <input></input> and <select></select>. CT has formalized representation of a tuple* $CT = \{name, type, id, value, location\}$.

Definition 2. ***Text Tag*** **(*TT*)**, *a string in HTML form, is used to prompt the content of input information associated with the control. TT has formalized representation of a tuple* $TT = \{name, location\}$.

Definition 3. ***Layout Tag*** **(*LT*)** *is used to illustrate the location of label in the form. They are mainly represented by <tr>, <td>, <th> and <table>, and so on. The purpose of layout tag is to fill location in Control Tag and Text Tag. Location is the*

coordinates that can record the row and column position of tags.

Users can actually see the query interface from a visual point of view, and it does not involve the internal code. We can divide query interface into three visual units according to three aspects: semantic interpretation (some label text), input value and constraint selection.

Definition 4. ***Text Unit* (*TU*)** *is usually seen as a label of query interface attribute, which is used to demonstrate the semantic meaning of the control value.*

Definition 5. ***Value Input Unit* (*VU*)** *refers to a text box and a drop-down list used for receiving user input or selecting a value.*

Definition 6. ***Constraint Selection Unit* (*CU*)** *is defined as a set of radio buttons or check boxes in query interface.*

Consider the presentation of attributes (a combination of *TU*, *VU* and *CU*), we define *Form Attribute* as follows:

Definition 7. ***Form Attribute* (*FA*),** *which shows the query capability of a query interface, has the corresponding TU for semantic interpretation, the VU for inputting query and CU for constraints. The formalized representation of FA in Equation (1) as follows:*

$$FA = \{TU, \{VU_i\}, \{CU_i\}\} \quad (1)$$

The set of *VU* may be empty, or have one or more elements. The set of *CU* only has two representations, one is empty, and the other one is the set only has one *CU*.

Many *Form Attributes* need to combine with each other by virtue of some semantic relationship. In this paper, we use domain ontology to guide the combination of text information in *FA* and define *Form Final Attribute* in query interface.

Definition 8. ***Form Final Attribute* (*FFA*)** *is a representation of combination between FAs, and the semantic relations between attributes guided by ontology. It can be formalized representation in Equation (2).*

$$FFA = \{O.c, \{FA_i\}, Rel\} \quad (2)$$

where $O.c$ is the related concepts in domain ontology, $\{FA_i\}$ is a set of *FA*s, Rel is the semantic relationship between each pair of $\{FA_i\}$.

Finally, we give a definition of *Interface Schema* which is used to automatically fill out the related information and then integrate a unified query interface.

Definition 9. ***Interface Schema* (*IS*)** *is not only a representation of general query interface forms and the result of extraction of query interface schema, but also the data processed by deep Web query interface integration for the follow-up work. It can be formalized representation in Equation (3).*

$$IS = \{url, action, method, name, \{FFA_i\}\} \quad (3)$$

where, *url* is the URL address of the interface, *action* is the address in which one program (usually a CGI script) deals with the form, *method* is the data transmission mode, which has two options, GET and POST, *name* is the internal name of the form.

Figure 1 shows the schema extraction framework of deep Web query interface. During the process of schema extraction, a detailed description contains five key steps, as described below.

• **Locating:** Based on some observations and statistical heuristic rules, we filter out the <form></form> segments that are not query interfaces, and try to locate the query interface in Web pages. Finally, we can get the internal code segments of query interface.

• **Parsing:** We obtain attribute information of each tag node in the query interface based on parsing the form. Parsing means reading valid data units in proper order and performing the corresponding process by identifying the feature of data unit.

• ***Form Attribute* Extraction:** Based on the visual character of layout and string similarity (Lin, Lyu, and King 2012) of tags and internal controls' names, we extract *Form Attribute* first time.

• ***Final Form Attribution* Extraction:** Many *Form Attributes* are not independent, and their text tags have semantic relations. It will be friendlier and have better semantics if we combine some *Form Attributes* who have semantic relations as a query condition. Through the guiding of domain ontology, we can add the semantic relations into the process of combining *Form Attribute*s and then get the *Final Form Attribute*.

• **Getting *Interface Schema*:** We get *Interface Schema* using the set of *Final Form Attribute* and the information of form such as *name*, *method* and *action*. Finally, the extraction of query interface schema is completed.

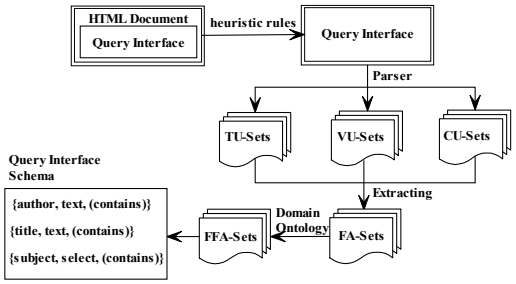

Figure 1: The schema extraction framework of deep Web query interface.

3 HOW TO INTEGRATE A UNIFIED INTERFACE

We now turn to the central issue of automatically discovering domain-specific information in deep Web, that is, integrating a unified ontology-based query interface on the basis of a two-phase matching method (direct matching and indirect matching), as described below.

Direct matching means that the elements in the form match the information in ontology based on keywords that refer to the words preprocessed and stored in word linked list. Direct matching process not only considers attribute tags, but also considers attribute values. Different specific domains have different representations of synonymous attributes, but they may have the same attribute value. So, the correct meaning of attributes can be inferred by the attribute value. Based on this inference, we can get more possible matches by attribute value after matching by attribute tags.

However, direct matching can not identify all the matchable concepts. Under this condition, indirect matching algorithm identifies matchable concepts by calculating the similarity between form attribute elements and ontology concepts. Because one concept may have several words, we describe two calculating methods: word similarity measure and concept similarity measure.

Word Similarity Measure: word similarity is calculated by common ontology *WordNet*. In *WordNet*, all concepts are represented as single words instead of phrases. If two words in *WordNet* are defined as similar relationship, then the similarity between them is 1. Otherwise, the similarity between them is calculated by Equation (4) below:

$$wdSim(w_1, w_2) = \frac{|p(w_1) \cap p(w_2)|}{SemOver(w_1, w_2)(|h(w_1) - h(w_2)| + 1)} \quad (4)$$

where, $|p(w_1) \cap p(w_2)|$ refers to semantic matching degree of words w_1 and w_2, $SemOver(w_1, w_2)$ is semantic distance of w_1 and w_2, $h(w)$ means depth of word w.

Concept Similarity Measure: each concept may consist of several words. Given two concepts: c_1 and c_2, for each word in c_1, the algorithm calculates the similarity between c_1 and each word in c_2 and stores the largest similarity. Equation (5) can calculate the concept similarity.

$$cnSimMid(c_i) = \frac{\sum_{k=1}^{n} wdSim(w_k, w_{k\_\max})}{n} \quad (5)$$

where $\sum_{k=1}^{n} wdSim(w_k, w_{k\_\max})$ refers to the largest similarity between word w_k in c_1 and the word in c_2. n means the number of words in c_1. Then the similarity of c_2 is calculated in the same way. We can get the final concept similarity with Equation (6) as follows.

$$cnSim(c_1, c_2) = \frac{cnSimMid(c_1) + cnSimMid(c_2)}{2} \quad (6)$$

Example 2. Assume that the similarity matrix ($CSM_{2\times2}$) of c_1 and c_2 is shown as follows:

$$CSM_{2\times2} = \begin{array}{c} \\ w_{21} \\ w_{22} \end{array} \begin{array}{c} \begin{array}{cc} w_{11} & w_{12} \end{array} \\ \begin{bmatrix} 0.19 & 0.25 \\ 0.33 & 0.55 \end{bmatrix} \end{array}$$

According to Equation (5), the similarity of c_1 and c_2 is described below:

$$cnSim\_mid(c_1) = ((wdSim(w_{11}, w_{22}) + wdSim(w_{12}, w_{22}))/2$$
$$= (0.33 + 0.55)/2 = 0.44$$
$$cnSim\_mid(c_2) = ((wdSim(w_{21}, w_{12}) + wdSim(w_{22}, w_{12}))/2$$
$$= (0.25 + 0.55)/2 = 0.4$$

Then according to Equation (6), the concept similarity between c_1 and c_2 is $cnSim(c_1, c_2) = (cnSim\_mid(c_1) + cnSim\_mid(c_2))/2$=(0.44+0.4)/2 =0.42.

4 AUTOMATIC QUERY TRANSFORMATION

In this section, we illustrate the function of query analyzer. The specific work of query transformer is shown in the next section. Before we describe the

specific query analyzing process, some basic concepts are given as follows:

Definition 10. ***Query Transformation*****.** Given the query Q_s in source query interface S and destination query interface T, the query set $\{Q_1, \ldots, Q_i\}(i = 1,2,\ldots)$ generated in T during the process of query transformation must satisfy (Zhang, He, and Chang 2004):

- Q_i must be an effective query in T.
- $Q_T = Q_1 \vee \ldots \vee Q_i$ must be as close to Qs as possible at semantic level in order to reduce the cost of filtering out the query results that do not meet source query requirements.

Next, the framework of query transformation is illustrated in Figure 2. And the specific procedures are described as follow.

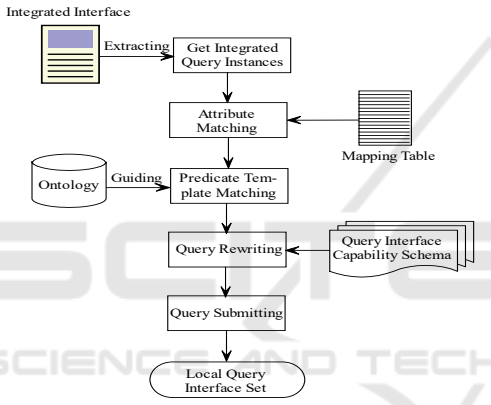

Figure 2: The framework of query transformation based on ontology.

Local interface schema is obtained by schema extractor. Under the guidance of ontology, we can get the matching relations between attributes. Then, an integrated form schema is generated. The relations between local interface attributes and integrated attributes can be recorded in mapping table at the same time. According to the information in mapping table, the query interface capability schema can be found mainly by exclusive attributes in forms. After users submit queries, we can get the value in database and store it in ontology. Meantime, the value input by users is stored in a global data structure as a source query Q_s delivered to the query transformer. Query transformation is divided into simple query transformation (1:1 matching) and complex query transformation (m:n matching). In order to reach the closest semantic meaning of query transformation, we design a type-driven minimum superset transformation algorithm based on ontology which designs different query transformation strategies according to the different types. And, the minimum superset query of destination predicate template is built under the guidance of ontology to make every transformation on predicate reach the closest semantic meaning. According to the obtained query model, the predicate templates that have been filled out are combined to one or more effective queries in local interfaces. An effective URL is submitted according to the attribute name and filling value transformed in each query form.

5 POST-PROCESSING QUERY RESULTS

Ranking query results includes ranking results that come from multiple databases and combining different results into an ordered result list which makes user feel that the results all come from the same database. The essence of ranking query results is computing the global similarity of each result and ranking the results based on the similarity. Global similarity cannot be obtained by local similarity because different databases have their own local calculation methods. Some similarity calculation methods are proposed in (Lu et al. 2005). In this paper, we rank the query results based on ontology similarity and the specific formula for each Web page record C is shown in Equation (7).

$$Q_k = \frac{t}{n} + Sim(C, O) \tag{7}$$

where n is the number of attributes in ontology. t is the number of attributes in one data record. The formula computes the weight Q_k for Web page k.

We take Barnes&Noble and Amazon as an example, suppose we search books about C language. There are eight attributes in the data record of Barnes&Noble and ten attributes in the data record of Amazon. $Sim(C,O)$ of the former is 0.085 and $Sim(C,O)$ of the latter is 0.093. So $Q_{Barnes\&Noble}$=8/12+0.085=0.752 and Q_{Amazon}=10/12+0.093 =0.926. Consequently, the content of Barnes&Noble will layout behind the content of Amazon because of $Q_{Barnes\&Noble} < Q_{Amazon}$.

Making use of this ranking algorithm, a snippet of ranking results is described in Figure 3 as follows.

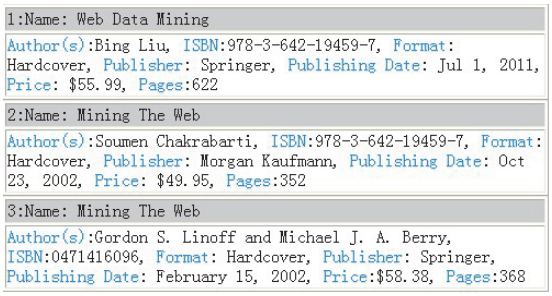

Figure 3: A snippet of result after ranking.

For the mutual benefit and protection of Authors and Publishers, it is necessary that Authors provide formal written Consent to Publish and Transfer of Copyright before publication of the Book. The signed Consent ensures that the publisher has the Author's authorization to publish the Contribution.

The copyright form is located on the authors' reserved area.

The form should be completed and signed by one author on behalf of all the other authors.

6 EXPERIMENTS AND RESULTS

There are many query integration methods. Among them, MetaQuerier is representative and outstanding from the perspective of effectiveness and efficiency. Thus, we select MetaQuerier as our baseline algorithm of interface integration. Based on domain ontology and common ontology *WordNet*, pattern matching is performed in the case that the number of interface forms is 25, 50, 75 and 100, respectively. And the comparison matching results are shown in Table 1.

Table 1 shows that most attributes can match in the two-phase pattern matching process which is based on ontology. Because of storing matching concepts timely and updating ontology continuously, the number of direct matching is much bigger than the number of indirect matching, which improves the accuracy and efficiency greatly.

Table 2 illustrates the comparison matching results on ten topics in order to reflect the comprehensive of the method.

Table 1: Interface matching results in domain book.

Table 1: Interface matching results in domain bookForm Number	Attribute Number	Successful Matching Number (ontology-based)		Successful Matching Number (Meta Querier)	Identified Matching Number		Precision (%)		Recall (%)		F-Measure (%)	
		Direct Matching Number	Indirect Matching Number		ontology	Meta Querier	ontology	Meta Querier	ontology	Meta Querier	ontology	Meta Querier
25	147	112	28	131	147	145	95.2	90.3	95.2	89.1	95.2	89.7
50	293	215	61	258	290	282	95.2	91.5	94.2	88.1	94.7	89.7
75	411	310	77	355	403	399	96.0	89.0	94.2	86.4	95.1	87.7
100	543	400	113	480	534	527	96.1	91.1	94.5	88.4	95.3	89.7

Table 2: Interface matching results on ten topics.

Topic	Form Number	Attribute Number	Successful Matching (ontology)		Successful Matching (Meta Querier)	Identified Matching Number		Precision (%)		Recall (%)		F-Measure (%)	
			Direct	Indirect		ontology	Meta Querier	ontology	Meta Querier	ontology	Meta Querier	ontology	Meta Querier
books	100	543	400	113	480	534	527	96.1	91.1	94.5	88.4	95.3	89.7
airfares	90	449	363	66	374	438	432	97.9	86.6	95.5	83.3	96.7	84.9
hotels	100	665	502	122	584	632	626	98.7	93.3	93.8	87.8	96.2	90.5
articles	100	504	337	154	444	498	492	98.6	90.2	97.4	88.1	98.0	89.2
computers	80	408	314	80	365	399	396	98.7	92.2	96.6	89.5	97.6	90.8
clothing	80	456	317	124	400	450	443	98.0	90.3	96.7	87.7	97.4	89.0
cell phones	100	590	440	135	528	583	580	98.6	91.0	97.5	89.5	98.0	90.3
music records	80	334	256	56	280	322	316	96.9	88.6	93.4	83.8	95.1	86.2
automobiles	100	342	257	60	282	322	318	98.4	88.7	92.7	82.5	95.5	85.5
movies	90	276	181	82	234	267	260	98.5	90.0	95.3	84.8	96.9	87.3

The method proposed by this paper not only identifies simple matching but also identifies complex matching. Table 3 shows the *precision* of simple matching and complex matching on different number of forms.

Table 3: Simple matching and complex matching results.

Form Number	Simple Matching(%)	Complex Matching(%)
25	96.6	83.5
50	97.2	82.4
75	97.8	83.8
100	98.6	85.3

Table 3 shows that both of them can achieve higher precision. However, it is harder to identify the structure of complex matching, so the precision of complex matching is a litter smaller.

Based on domain ontology and common ontology *WordNet*, query transformation is performed in the case that the number of interface forms is 25, 50 and 75, respectively. And the comparison transforming results using ontology and NFB are shown in Table 4. Table 5 illustrates the transforming results on ten topics to reflect the comprehensive of this method.

To evaluate the feasibility and efficiency of our post-processing approach, several tests have been conducted to test our system. Table 6 illustrates the extracting results over 4 websites in order to reflect the comprehensive of the method.

7 CONCLUSIONS

This paper presents an automatically discovering system for retrieving resources across multiple databases that are accessible in hidden Web. We provide a comprehensive schema extraction sub-system for extracting the interface schema, together with interface integration sub-system for integrating a unified query interface and query transformation sub-system for transforming a query in the integrated form to multiple queries in domain-specific interfaces. The schema extraction technique involves two perspectives, internal code and visual unit, which serve as the foundation for integrating a unified query interface. In the integration process, finding a keyword-based matching concept is straightforward.

Table 4: The query transformation results in the case of different number of input forms using ontology and NFB.

Form Number	Relevant Query Transform Templates	All the Templates Transformed		All the Templates Transformed Correctly		Precision (%)		Recall (%)		F-measure (%)	
		Ontology	NFB	Ontology	NFB	Ontology	NFB	Ontology	NFB	Ontology	NFB
25	146	139	141	129	123	92.8	87.2	88.4	84.2	90.5	85.7
50	284	269	260	247	232	91.8	89.2	87.0	81.7	89.3	85.3
75	407	382	364	354	320	92.7	87.9	87.0	78.6	89.8	83.0

Table 5: The transforming results on ten topics.

Topic	Form Number	Relevant Query Transform Templates	All the templates transformed		All the templates transformed correctly		Precision (%)		Recall (%)		F-measure (%)	
books	75	407	382	338	354	324	92.7	95.9	87.0	79.6	89.8	87.0
airfares	90	437	415	386	374	349	90.1	90.4	85.6	79.9	87.8	84.8
hotels	100	494	479	460	435	404	90.8	87.8	88.1	81.8	89.4	84.7
articles	100	658	637	613	616	582	96.7	94.9	93.6	88.4	95.1	91.5
computers	80	387	373	357	342	322	91.7	90.2	88.4	83.2	90.0	86.6
clothing	80	436	421	404	398	361	94.5	89.4	91.3	82.8	92.9	86.0
cell phones	100	575	557	533	538	507	96.6	95.1	93.6	88.2	95.1	91.5
music records	80	327	313	299	287	263	91.7	88.0	87.8	80.4	89.7	84.0
automobiles	100	340	318	307	299	280	94.0	91.2	87.9	82.4	90.8	86.6
movies	90	265	256	212	232	195	90.6	92.0	87.5	73.6	89.0	81.8

Table 6: The extracting results over four different websites.

Website	Test Number	Relevant Number	Extracted Number	Correct Extracted Number	Precision (%)	Recall (%)	F-Measure (%)
www.barnesandnoble.com	158	154	150	143	95.3	92.9	94.1
www.amazon.com	189	180	174	168	96.6	93.3	94.9
www.chaucersbooks.com	137	129	123	118	95.9	91.5	93.7
www.chapters.indigo.ca	147	139	132	124	93.9	89.2	91.5

For the concepts that direct matching process can not identify, we design a two-phase matching method that calculating the semantic similarity based on Levenshtein Distance and *WordNet*. We also present a type-driven minimum superset query transformation method utilized interface capability schema to rewrite queries. Beside, a creative strategy (null correspondence) is applied to improve the efficiency of transforming. Some irrelative parts such as ads, navigation and flash, are get rid of in the process of reducing noisy. Basic information is extracted to simplify final results. Finally, we propose a ranking algorithm to present users an ordered list of data records. Our technique is efficient and scalable. A series of experiments show that the method proposed in this paper is both accurate and feasible, which could significantly and semantically improve and enhance the hidden Web resources discovering.

REFERENCES

An, Y. J.; Chun, S. A.; Huang, K. C.; and Geller, J. 2008. Enriching Ontology for Deep Web Search. Lecture Notes in *Computer Science. 5181: 73-80*.

Fan, W. F.; Gao, H.; Jia, X. B.; Li, J. Z.; and Ma, S. 2011. Dynamic constraints for record matching. *VLDB Journal. 20(4):495-520*.

Shestakov, D.; Bhowmick, S.; and Lim, E. P. 2005. DEQUE: Querying the Deep Web. *Data and Knowledge Engineering. 52(3): 273-311*.

He, B.; Chang, K. C. C.; and Han, J. W. 2004. Discovering Complex Matchings across Web Query Interfaces: A Correlation Mining Approach. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 148–157. Seattle, WA*.

He, H.; Meng, W. Y.; and Lu, Y. Y. 2007. Towards Deeper Understanding of the Search Interfaces of the Deep Web. *Word Wide Web Journal. 10 (2):133-155*.

He, H.; Meng, W. Y.; and Yu, C. T. 2005. Constructing interface schemas for search interfaces of Web databases. In *Proceedings of the 6th International Conference on Web Information Systems Engineering, 29-42. New York*.

He, H.; Meng, W. Y.; Yu, C. T.; and Wu, Z. 2003. WISE-Integrator: An Automatic Integrator of Web Search Interfaces for E-commerce. In *Proceedings of the 29th International Conference on Very Large Data Bases, 357-368. Berlin*.

He, Z. T.; Hong, J.; and Bell, D. A. 2009. A Prioritized Collective Selection Strategy for Schema Matching across Query Interfaces. In *Proceedings of the 26th British national conference on Databases, LNCS 5588. 21-32*.

Hong, J.; He, Z. T.; and Bell, D. A. 2009. Extracting Web Query Interfaces Based on Form Structures and Semantic Similarity. *IEEE 25th International Conference on Data Engineering. 1-3:1259-1262*.

Lin, Z.J .; Lyu, M. R.; and King, I. 2012. MatchSim: a novel similarity measure based on maximum neighborhood matching. *Knowledge and Information Systems. 32(1):141-166*.

Liu, W.; Meng, X. F.; and Meng, W. Y. 2010. ViDE: A Vision-Based Approach for Deep Web Data Extraction. *IEEE transactions on knowledge and data engineering. 22(3): 447-460*.

Lu, Y. Y.; Meng, W. Y.; Shu, L. C.; Yu, C.; and Liu, K. L. 2005. Evaluation of Result Merging Strategies for Metasearch Engines. *6th International Conference on Web Information Systems Engineering, 53-66. New York*.

Madhavan, J.; Jeffery, S. R.; Cohen, S. I.; Dong, X.; Ko, D.; and Yu, C. 2007. Web-scale data integration: You can only afford to pay as you go. In *Proceedings of Conference on Innovative Data Systems Research, 40-48. New York, NY.:ACMPress*.

Zhang, Z.; He, B.; Chang, K. C. C. 2004. On-the-fly constraint mapping across Web query interfaces. In *Proceedings of the VLDB Workshop on Information Integration on the Web, 1-6. Toront*.