[Downloaded from www.aece.ro on Thursday, March 28, 2024 at 14:58:47 (UTC) by 3.235.139.122. Redistribution subject to AECE license or copyright.]

Post-processing of Deep Web Information Extraction Based on Domain Ontology

Lu LIU[1,2], Tao PENG[1,2,3*]
[1]*College of Computer Science and Technology, Jilin University, Changchun 130012, China*
[2]*Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA*
[3]*Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun 130012, China*
**Corresponding author:taopeng@illinois.edu,tpeng@jlu.edu.cn*

***Abstract*—Many methods are utilized to extract and process query results in deep Web, which rely on the different structures of Web pages and various designing modes of databases. However, some semantic meanings and relations are ignored. So, in this paper, we present an approach for post-processing deep Web query results based on domain ontology which can utilize the semantic meanings and relations. A *block identification model* (*BIM*) based on node similarity is defined to extract data blocks that are relevant to specific domain after reducing noisy nodes. Feature vector of domain books is obtained by *result set extraction model* (*RSEM*) based on *vector space model* (*VSM*). *RSEM*, in combination with *BIM*, builds the domain ontology on books which can not only remove the limit of Web page structures when extracting data information, but also make use of semantic meanings of domain ontology. After extracting basic information of Web pages, a ranking algorithm is adopted to offer an ordered list of data records to users. Experimental results show that *BIM* and *RSEM* extract data blocks and build domain ontology accurately. In addition, relevant data records and basic information are extracted and ranked. The performances *precision* and *recall* show that our proposed method is feasible and efficient.**

***Index Terms*—Knowledge based systems, Machine learning, Semantic Web, Web mining, World Wide Web.**

I. INTRODUCTION

With the advent of information technology and development of network technology, people are capable of obtaining abundant information from WWW just through the conventional search engine. Users only need to fill the query interface in HTML pages, and then the request is submitted to Web server. To respond users, the Web servers, which contain multiple databases, index the corresponding information and directly return it to users. Nowadays, the total Web is almost made up of two parts. One is a collection of hyperlinks (shallow Web) composed of static pages, which can be indexed by a traditional search engine. However, many online network databases, which have a larger amount of data, are hidden behind the query interface in the Web page. So another one is hidden Web (deep Web) which is accessed and gets resources by posting queries to the query interface provided in the website, instead of specifying a URL to send an HTTP request to get the static page information.

In addition, extracting domain-specific information is of much more significance. However, the different styles and structures of Web pages and databases, and advertisement or some information that users do not interested in, all bother users to search the Internet. To tackle these problems, we mainly extract contents from Web pages on domain books returned by search engine [1]. So far, there are many information extraction techniques such as natural-language-based, XML-based and DOM-based extraction techniques, which are described below. Their performances all vary from the Web page structures. It is very trouble-some for extracting information correctly. This paper proposes an extraction method based on domain ontology which is without restriction of Web page structures.

In order to facilitate users searching the Web, domain-specific information or advertisements are demonstrated using data records (shown in Fig.1). All data records group into data region. The data records need to be extracted from multiple databases which are designed using different styles. Also, they need to be organized, simplified and converted into a computer-readable format. Because most Web pages are written in HTML language which makes fully use of tags to layout the Web pages. 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 relative data blocks in HTML tree are extracted, ranked and stored in our database which has a unique storage structure and stores basic information on domain books. The first advantage of our proposed method is it could extract domain-specific information without any useless text. What's more, it could get rid of the restriction of Web page struction. Finally, it uses data records to demonstrate domain-specific information which could facilitate users searching the Web.

The outline of this paper is organized as follows. We review the related work in Section II. Section III illustrates how to extract query results and build domain ontology based on block identification model and *RESM*. Section IV proposes the process of post-processing extracted query results including how to rank them. Several comprehensive experiments are performed to evaluate the effectiveness and efficiency of our method in Section V. Section VI draws the conclusions.

II. RELATED WORK

Query results extraction aims at extracting information from result pages, processing unstructured texts that including valuable information and organizing them into

Digital Object Identifier 10.4316/AECE.2013.04005

1582-7445 © 2013 AECE

[Downloaded from www.aece.ro on Thursday, March 28, 2024 at 14:58:47 (UTC) by 3.235.139.122. Redistribution subject to AECE license or copyright.]

structured textual information. Different Web pages format their information in their own ways which makes it difficult to extract relevant data [2]. The idea presented in this paper is a new method. Before post-processing deep Web query results, these results should be extracted, integrated, and then transformed. After this, the results could be post-processed based on domain ontology. So, in what follows, this paper mainly introduces some related work about the previous research, such as how to extract, integrate, and transform deep Web query results.

The Message Understanding Conferences (MUCs) [3], which were designed to promote and evaluate research in information extraction, were initiated by NOSC (Naval Ocean Systems Center) to assess and to foster research on the automated analysis of military messages containing textual information. Gregg et al. [4] presented an adaptive information extraction system prototype that combined multiple information extraction approaches to allow more accurate and resilient data extraction for a widely variety of Web resources. He et al. [5-7] presented a research project on database integration called *DMSE-Web*, and developed an extraction tool called *WISE-IExtractor* to get query interface schema. Peng et al. [8] presented an attempt to process semantic queries against the spatial database and demonstrated the function of spatial semantic queries via a practical prototype system. Doorenbos et al. [9] made use of heuristic information about a specific domain to fill the forms and developed *ShopBot*, a fully-implemented, domain-independent comparison-shopping agent. Given the home pages of several online stores, *ShopBot* automatically learns how to shop at those vendors. Ipeirotis et al. [10] developed *QProber*, a system that automatically categorized hidden Web text databases in a classification scheme, according to their topical focus. *QProber* used just the number of matches generated from a small number of topically focused query probes. Ashraf et al. [11] proposed a system that employed clustering techniques for automatic information extraction from HTML documents containing semi-structured data. Liu et al [12] proposed a method for automatic extraction of structured data from Web pages and built a tag tree based on visual information. This method will also ignore some semantic information in the process of extracting. Hong [13] used automatic wrapper to extract three types of data records (single-section, multiple section and loosely structured data records) from deep Web.

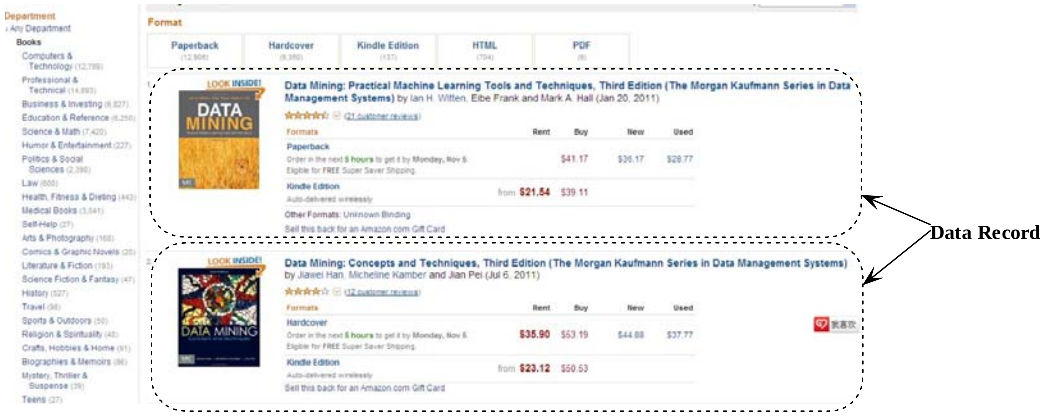

Department
Any Department
Books
Format
Paperback
Hardcover
Kindle Edition
HTML
PDF
Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (The Morgan Kaufmann Series in Data Management Systems) by Ian H. Witten, Eibe Frank and Mark A. Hall (Jan 20, 2011)
Data Mining: Concepts and Techniques, Third Edition (The Morgan Kaufmann Series in Data Management Systems) by Jiawei Han, Micheline Kamber and Jian Pei (Jul 6, 2011)
Data Record

Figure 1. An example of data records on domain books.

III. DOMAIN ONTOLOGY CONSTRUCTION BASED ON QUERY RESULT EXTRACTION

A. *The comparison of different query result extraction methods*

So far, many methods are used to extract query results such as extraction techniques based on natural language, HTML, wrapper [14], Web, XML [15], DOM tree and so on. Natural-language-based extraction technique mainly deals with text documents and needs an informative knowledge base to support the whole system. For example, WHISK [16] is a typical natural-language-based system, which divides one text into multiple basic elements and returns a group of elements to users every time. Then users mark some information that they are interested in, so that the system can accomplish the extraction task according to extraction rules generated through syntactic analysis, semantic analysis and a series of reasoning analysis. However, this method can only be applied in specific domain and can not handle the text documents that do not conform to the grammar structure. Transplanting the system into another domain will need a lot of manual work to rebuild the system. RoadRunner [17] is a representative extraction system based on DOM. As we known, most Web pages are written in HTML, and they are extensible and platform-independent. For example, if users want to filter the spam, that is, prune the DOM tree, then we tidy the HTML page into a well-formed Web page using HTML TIDY tool (http://www.w3.org/People/Raggett/tidy/). Once the similarity between a under-extracted Web page and a given Web page template is greater than a specific threshold, the Web page will be extracted and vice versa. View-based extraction method is achieved mainly by making fully use of page structure, such as Vision-based Page Segmentation [18]. View Segmentation algorithm is utilized to partition a Web page into several sub-parts. A series algorithms and rules are applied in the last data block to extract some useful information. Ontology-based extraction method performs according to the data and the description information, and rarely relies on the Web page structure. In addition, ontology-based extraction method can add semantic meanings and relations during the process of extraction, which makes the extracting results more accurate. Only if the ontology is informative enough, it can extract a variety of text documents and overcome the weakness above such as relying on Web page structure (DOM-based) and abundant manual work (natural-language-

[Downloaded from www.aece.ro on Thursday, March 28, 2024 at 14:58:47 (UTC) by 3.235.139.122. Redistribution subject to AECE license or copyright.]

based). So, in this paper, we build our own ontology and take book domain as an example to extract information.

B. Potential data block identification

Generally, a query-result Web page may contain many data blocks such as advertisements, data information, and navigation and so on. However, users usually need to know the content of the data information part instead of some noisy parts that disturb users to get the data information directly. Accordingly, locating and identifying the data information in Web pages exactly are essential to process the query results. By now, we can identify the potential data block through many methods. One is page structure method. It works by considering that the data information needed by users often locates in the center of the whole Web page. The other one is space ratio method [19]. This method holds that the data should account for the largest area of the whole Web page. The two methods above both locate the data blocks relying on the visual effects, which vary from the Web page structure, so they do not possess good computation feasibility and credibility.

In order to overcome the weakness above, this paper presents a *Block Identification Model* based on node similarity to identify the potential data block that possibly contains query results. Before locating data block, under-analyzed Web pages need to be parsed by HTML parser to get rid of noisy nodes, which are useless, so that we can improve the efficiency. The worthless tags that are not useful for extracting information include <script>, <style>, <font>, <img>, <select>, <input>, namespace tags, navigation tags and those tags whose contents are null and types are "hidden". *Breadth First Search* [20] is adopted to reduce noisy nodes in DOM tree (shown in Fig.2) and the specific procedure is described as follows:

Step 1. Search DOM tree using *BFS* from the root.

Step 2. Judge whether the node is "body" node. If yes, jump to *Step 3*, otherwise, continue searching.

Step 3. Read nodes one by one. Judge whether it reaches the end of the tree. If yes, the algorithm ends, otherwise, jump to *Step 4*.

Step 4. Judge whether the node that is being searched is noisy node. If yes, delete it and its sub-tree, otherwise, jump to *Step 3*.

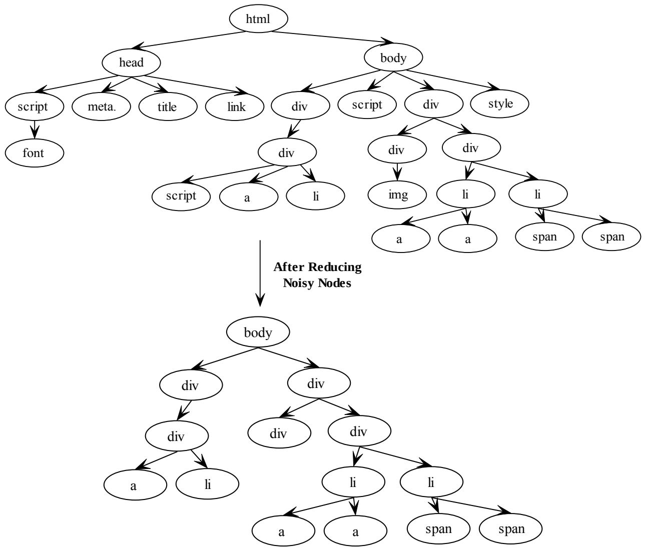


Figure 2. The illustration of reducing noisy nodes.

In the *BIM*, if two nodes are made up of more structure-similar sub-trees, the data region that formed by generated nodes is more likely to be the one that we want to search for [21]. We compute the similarity between sub-trees generated by the same node *T*. If the similarity is greater than a threshold, then the data blocks generated by *T* are likely to be the query result block. The similarity between sub-tree T_1 and T_2 is calculated by Eq.1 as follows:

$$Sim(T_1,T_2)=\frac{sumT_1+sumT_2-disumT_1T_2}{sumT_1+sumT_2} \qquad (1)$$

where $sumT_1$, $sumT_2$ and $disumT_1T_2$ are three variables that are adopted to compute the similarity between sub-trees. And they choose different calculation method according to different circumstances described as follows:

1. There are two nodes located in the corresponding position in DOM trees, and the tags are the same. Then $sumT_1= sumT_1+1$; $sumT_2=sumT_2+1$.
2. There are two nodes located in the corresponding position in DOM trees, but the tags are different. Then $sumT_1= sumT_1+1$; $sumT_2=sumT_2+1$; $disumT_1T_2= disumT_1T_2+1$.
3. There is a node located in the corresponding position of T_1, but there is no node existing in the corresponding position of T_2. Then $sumT_1= sumT_1+1$; $disumT_1T_2= disumT_1T_2+1$.
4. There is a node located in the corresponding position of T_2, but there is no node existing in the corresponding position of T_1. Then $sumT_2=sumT_2+1$; $disumT_1T_2= disumT_1T_2+1$.

Through the calculation procedure above, in the DOM tree, all minimum sub-trees that include similar nodes can be identified. For example, given a DOM tree (as shown in Fig.3) and suppose we want to compute the similarity between sub-trees T_1 and T_2. According to our method, $sumT_1=6$, $sumT_2=5$, $disumT_1T_2=2$, the similarity between sub-trees T_1 and T_2 is calculated by Eq.1 and $Sim(T_1, T_2)=(6+5-2)/(6+5)=0.818$.

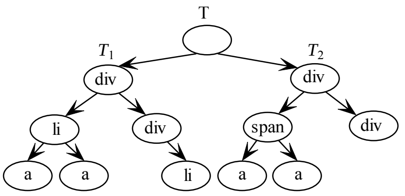


Figure 3. An example of DOM tree for computing similarity between sub-trees.

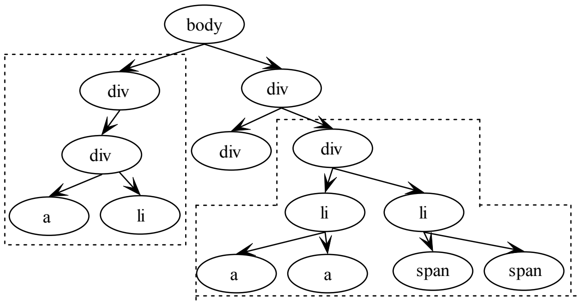


Figure 4. An illustration of the potential data blocks.

To identify the potential data blocks, we set up a threshold θ to judge whether the data block formed by a sub-tree is the domain-specific data block that contains valuable data information. If $Sim(T_1, T_2)>\theta$, we consider the data block to be the one that we want to extract (as shown in

[Downloaded from www.aece.ro on Thursday, March 28, 2024 at 14:58:47 (UTC) by 3.235.139.122. Redistribution subject to AECE license or copyright.]

the rectangle of Fig.4). The domain-specific query results can be retrieved by applying our classification method to identify the domain-specific sub-trees which will be described in the following sections.

C. Result set extraction model

In this section, we present a *result set extraction model* (*RSEM*) based on *Vector Space Model* (*VSM*). The architecture of *RSEM* is shown in Fig.5 as follows. 50 Web pages including query results on domain books are analyzed manually to extract feature vector which is shown in TABLE I.

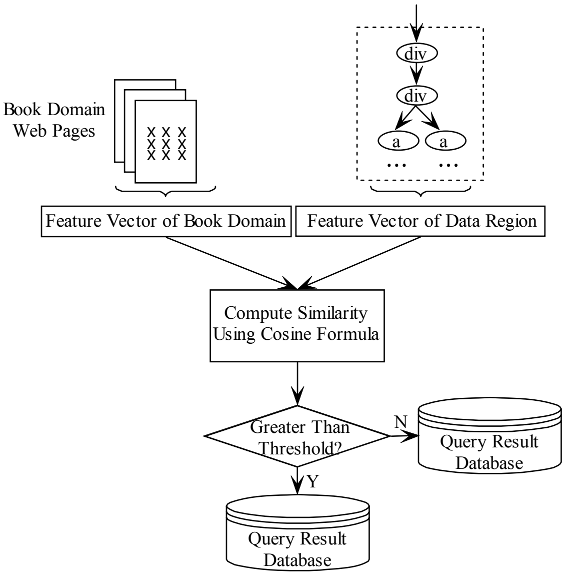


Figure 5. The architecture of *RESM*.

TABLE I. FEATURE VECTOR TABLE OF BOOK DOMAIN.

Feature	Book Name	Author	Publisher	ISBN
Weight	0.25	0.19	0.17	0.14
Feature	Publishing Date	Price	Total Pages	Edition
Weight	0.13	0.06	0.04	0.02

The weight of each feature is calculated by Eq.2 as follows.

$$r_i = \frac{t_i}{T} \tag{2}$$

where t_i is the occurrence times of the i^{th} feature in all data records of 50 Web pages. T is the total times of all attributes appeared in 50 Web pages. Besides, we use this formula based on a hypothesis that each attribute only appears once in each data record. So, the weight sum of all features will be 1.

In order to obtain the data blocks that are most similar to the feature vector, a host of methods for calculating similarity have been proposed and the most successful of these is *cosine correlation* method of *VSM* [22]. In *VSM*, suppose that data blocks are part of an *m*-dimensional vector space, where *m* is the number of features such as word, phase and so on. The *m*-dimensional vector of a data block *d* is represented as $d= (d_1, d_2, \dots , d_m)$ and the *m*-dimensional feature vector of domain books (as shown in TABLE I) is represented as $v= (v_1, v_2, \dots , v_m)$, so that we can compute the similarity by *cosine correlation* in Eq.3 as follows:

$$Cosine(\bar{d},\bar{v}) = \frac{\sum_{k=1}^{m}(d_k \cdot v_k)}{\sqrt{\sum_{k=1}^{m} d_k^{\,2}} * \sqrt{\sum_{k=1}^{m} v_k^{\,2}}} \tag{3}$$

It can be seen from the formula: the cosine value between two identical vectors will be 1 because the angle is zero, and the cosine value will be zero if two vectors do not share any non-zero features. Once the similarity between the data block and the feature vector of TABLE I is greater than a threshold λ , the data block is regarded as a query result record. The transformation procedure is shown in Fig.6.

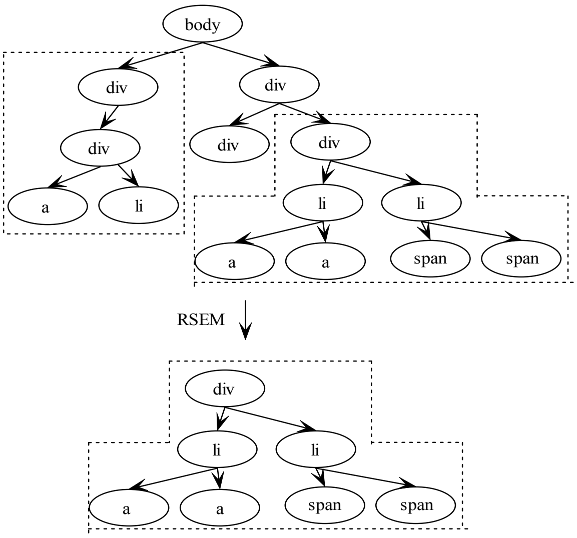


Figure 6. The transformation procedure of *RSEM*.

D. Domain ontology construction based on RSEM

The goal of constructing domain ontology is taking advantage of the semantic meanings and relations in ontology, which makes the extraction process more accurate. Besides, ontology-based extraction method can get rid of the restriction of different Web page structures. Our ontology construction on domain books is represented by a ontology conceptual model which is described as a tuple.

$$Class=\{CC, DT, \{S_i\}, \{CI_i\}, \{CA_i\}, \{SC_i\}, \{n_i\}, N\}.$$

where *CC* is the core class which can be defined as the key words of a concept. *DT* is the data type such as “string”, “int” and so on. $\{S_i\}$ is the set of synonyms of *CC*, that is, the alias name of the concept. $\{CI_i\}$ is the set of instances of core class. $\{CA_i\}$ is the set of attributes that correspond to a concept. $\{SC_i\}$ is the set of subclasses of core class. $\{n_i\}$ is the number of ontology instances. *N* is the number of core class in ontology.

The ontology conceptual model developed by Protégé [23] consists two parts (attribute model and attribute description) to make it be identified and processed by computer. Attribute model describes the organization structure of entities in the same domain. For example, Fig.7 shows the attribute model on domain books. The book in the center of Fig.7 represents the entity in the domain and the six branches represent the attributes of the entity (title, author(s), publishing date, price, ISBN, publisher). Attribute description lists the attribute values belonging to the attribute domain. For example, Fig.8 demonstrates the description of attribute “price”. *Name* means the name of

[Downloaded from www.aece.ro on Thursday, March 28, 2024 at 14:58:47 (UTC) by 3.235.139.122. Redistribution subject to AECE license or copyright.]

attribute itself such as “price”. *Alias names* represent the names that can replace the attribute name such as “our price”, “buy”, “new”, “used”, “minimum price”. *Data type* refers to the type of the attribute. In this paper, data types include “time”, “integer”, “real”, “price” and “character”. *Data Value* means the specific values that have shown up in the training set of query result pages such as “$55.38”, “49.85$”. *Value possibility* is the similarity between values in the result records. *Maximum Occurrence Number* refers to the greatest times that an attribute value shows in the training set of query result pages.

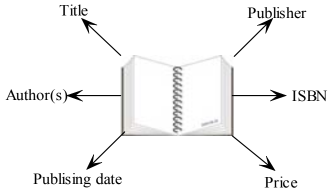


Figure 7. Attribute model.

Name: Price
Alias Names: our price, buy, new, used, minimum price
Data Type: price
Data Value: $55.38, 49.85$
Value Possibility: 85%
Name Possibility: 88%
Maximum Occurence Number: 2

Figure 8. Attribute description of price.

The query result record is mapped into vector space based on *RSEM* to compute the similarity with feature vector so that the eligible query result records can be extracted. A combination segment of extract results is shown in TABLE II as follows.

Before domain ontology on books is built, query result annotation and attribute match need to be conducted. Query result annotation refers to using explicit words to annotate all data information in data result record. Four heuristic rules are adopted to guide the process of annotation as follows.

Rule 1. If some keywords often appear in a specific column of query result form, then the tag name of the keyword should be the annotation. Because some query interfaces on HTML Web pages contain some data attributes and Web page designers will focus on using the most relevant data to respond users’ queries, the keywords submitted in the query interface will also show up in the query result set.

Rule 2. Search head tags in tabular HTML pages. In order to facilitate attaching title in each column of each HTML form, some tags such as <TH>, <THEAD> are defined in HTML in detail. These tags often show up around the data information in the query result Web pages. Therefore, head tags like <TH>, <THEAD> need to be checked around data values. And once *Rule 1* does not work, the title is regarded as the annotation in query result records.

Rule 3. Combine tags and data information. Because they often combine with each other in HTML pages. For example, authors(s), publishing date, price, ISBN and publisher are the common prefixes in each column in TABLE II. When *Rule 1* and *Rule 2* both do not work, *Rule 3* is selected to annotate.

Rule 4. Name annotation using conventional format or representative symbol. For example, we use “month date, year” format to represent date and symbol “$” to represent price.

Besides, each rule has its priority (*Rule 1*>*Rule 2* >*Rule 3* > *Rule 4*). When they face with conflicts, annotation is made according to their priorities. TABLE III is the annotation result of TABLE II achieved by the four heuristic rules above.

TABLE II. A COMBINATION SEGMENT OF EXTRACTION RESULTS.

	Author(s)	Publishing date	Price	ISBN	Publisher
Web Data Mining	Bing Liu	Jul 1,2011	$55.99	978-3-642-19459-7	Springer
	Author(s)	Publishing date	Price	ISBN	Publisher
Mining The Web	Soumen Chakrabarti	Oct 23, 2002	$49.95	978-1-55860-754-5	Morgan Kaufmann
	Author(s)	Publishing date	Price	ISBN	Publisher
Mining The Web	Gordon S. Linoff and Michael J. A. Berry	February 15, 2002	$58.38	0471416096	Wiley

TABLE III. ANNOTATION RESULT OF TABLE II.

Title	Author(s)	Publishing date	Price	ISBN	Publisher
Web Data Mining	Bing Liu	Jul 1,2011	$55.99	978-3-642-19459-7	Springer
Mining The Web	Soumen Chakrabarti	Oct 23, 2002	$49.95	978-1-55860-754-5	Morgan Kaufmann
Mining The Web	Gordon S. Linoff and Michael J. A. Berry	February 15, 2002	$58.38	0471416096	Wiley

The attributes from one book website are identical. However, they are not identical when coming from different websites. For example, some websites only have attribute value such as author’s name but do not have tags. So the corresponding annotation words need to be got from some attributes that have been annotated if no correct annotation can be found. The whole process is achieved by attribute match introduced in [24].

IV. POST-PROCESSING OF DEEP WEB QUERY RESULTS

A. *Extracting query results based on domain ontology*

In the section above, *BIM* was presented to identify the potential data blocks that may contain query results in deep Web, in combination with *RSEM* to extract query result record on domain books. According to the two models and the extracted records, domain ontology was built and

[Downloaded from www.aece.ro on Thursday, March 28, 2024 at 14:58:47 (UTC) by 3.235.139.122. Redistribution subject to AECE license or copyright.]

learning process was achieved. In what follows, algorithm of query result extraction will be described based on domain ontology. Besides, result records will be ranked using a ranking algorithm and then offer to users. Both of the two algorithms are made up of post-processing in deep Web. The architecture of post-processing is shown in Fig.9 as follows.

Query result pages are made up of a group of query result records. After parsing, Web page record $C = \{c_1, c_2, \ldots, c_n\}$ can be judged whether it is our needed information. Each c_i is assigned with its corresponding weight w_i which is defined in Eq. 4 as follows:

$$w_i = \begin{cases} p_{aj}, & \text{if } c_i \text{ is the name or alias of attribute } a_j \\ w_j, & \text{otherwise, it is equal to the most likely attribute possibility} \end{cases} \qquad (4)$$

where p_{aj} is the name possibility of attribute a_j which is described in Fig.8. The similarity between domain ontology and the Web page record is calculated by Eq.5 below.

$$Sim(C,O) = \frac{\sum_{i=1}^{n} w_i}{mn} \qquad (5)$$

where O means ontology. m is the number of attributes in domain ontology on books (m=12 in this paper). n is the number of attributes in one data record in a Web page which varies from the different websites.

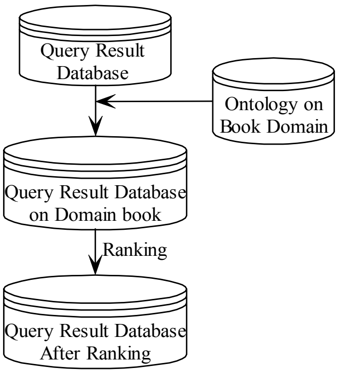

Query Result Database
Ontology on Book Domain
Query Result Database on Domain book
Ranking
Query Result Database After Ranking

Figure 9. Architecture of post-processing.

Given a root node R in HTML tree after parsing and reducing noisy nodes and a variable t which records the number of attributes in HTML that are identical with those in ontology. The algorithm of extracting basic information on domain books is described as follows.

```
Algorithm of Extracting Basic Information
Input: $O$: Query result after parsing ontology. $R$: an HTML tree of under-extracted Web page after processing
Output: Basic Information on domain books

procedure Extracting Basic Information
  $Tp \leftarrow R$; $i \leftarrow 1$
  repeat
    $Tp_i \leftarrow$ a child node that has the biggest similarity with ontology $O$
    if $\Delta$ is greater than a threshold $\tau$ and $Sim(Tp_i, O) > Sim(Tp, O)$ then
      $Tp \leftarrow Tp_i$
    else
      break
    end if
  until $Tp$ reaches its leaf node
  repeat
    $Tp_l \leftarrow$ the left brother node of $Tp$
    if $\Delta$ is greater than a threshold $\tau$ and $|Sim(Tp+Tp_l,O) - Sim(Tp, O)| < \varepsilon$
    then
      $Tp \leftarrow Tp+Tp_l$
      $i \leftarrow i+1$
    else
      break
  until $Tp_l$ reaches its leaf node
  repeat
    $Tp_r \leftarrow$ the right brother node of $Tp$
    if $\Delta$ is greater than a threshold $\tau$ and $|Sim(Tp+Tp_r,O) - Sim(Tp, O)| < \varepsilon$
    then
      $Tp \leftarrow Tp+Tp_r$
      $i \leftarrow i+1$
    else
      break
  until $Tp_r$ reaches its leaf node
  $t \leftarrow i$
end procedure
```

Algorithm 1. Extracting basic information on domain books. $\Delta = aSimAuthor + bSimPrice + cSimDate + dSimISBN$. *SimAuthor* (*SimPrice*, *SimDate* and *SimISBN*) refers to the similarity between the author (price, date and ISBN) in ontology and the corresponding attribute in under-extracted Web page and it should be computed before extracting basic information. $a+b+c+d$ =1. Adjusting a,b,c and d can control the importance of each attribute.

Double thresholds are adopted in the algorithm above. First, we continually find the most similar root of a sub-tree with the ontology. If there is no eligible root, it means the Web page does not contain information that we want. Otherwise, we start to judge whether its sibling nodes are instances that we want to extract using threshold τ. Next, the difference of similarity before inserting sibling nodes and after inserting sibling nodes should be controlled in a certain range ε. The searching process is performed repeatedly until no eligible nodes can be found.

B. Ranking 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 can not be obtained by local similarity because different databases have their own local calculation methods. Some similarity calculation methods are proposed in [25]. 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 Eq.6 as follows.

$$Q_k = \frac{t}{n} + Sim(C,O) \qquad (6)$$

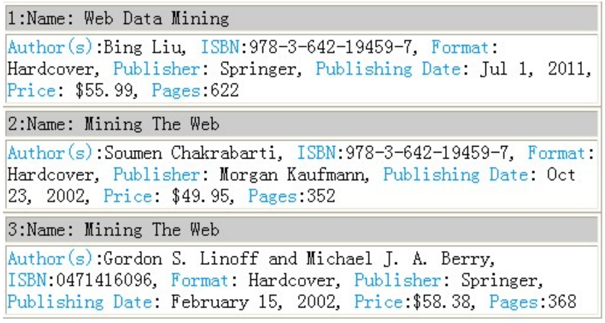
where n is the number of attributes in ontology. t is the number of attributes in one data record. The formula computes the weight Q_k for Web page k.

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

[Downloaded from www.aece.ro on Thursday, March 28, 2024 at 14:58:47 (UTC) by 3.235.139.122. Redistribution subject to AECE license or copyright.]

results is described in Fig.10.



1:Name: Web Data Mining

Author(s):Bing Liu, ISBN:978-3-642-19459-7, Format: Hardcover, Publisher: Springer, Publishing Date: Jul 1, 2011, Price: $55.99, Pages:622

2:Name: Mining The Web

Author(s):Soumen Chakrabarti, ISBN:978-3-642-19459-7, Format: Hardcover, Publisher: Morgan Kaufmann, Publishing Date: Oct 23, 2002, Price: $49.95, Pages:352

3:Name: Mining The Web

Author(s):Gordon S. Linoff and Michael J. A. Berry, ISBN:0471416096, Format: Hardcover, Publisher: Springer, Publishing Date: February 15, 2002, Price:$58.38, Pages:368

Figure 10. A snippet of result after ranking.

V. EXPERIMENTS AND RESULTS

To evaluate the feasibility and efficiency of our approach, several tests have been conducted to test our system. In the experiment, two models are used to extract data records on domain books from multiple databases covering hundred of data records. Under the guidance of our method, basic information about domain books are extracted and organized according to the strategies of deep Web post-processing.

A. Performance metrics

The two most common effectiveness measures, *precision* and *recall*, were introduced to summarize and compare search results. Intuitively, *precision* is the fraction of extracted data records that are relevant to the domain books, which measures how well it is doing at rejecting irrelevant data records. *recall* is the fraction of extracted data records which are relevant indeed, and measures how well it is doing at finding all the relevant data records. However, the relevant set for any given topic is unknown in the Web, so the true *recall* is hardly to measure. In view of this situation, we delineate a specific network, which is regard as a virtual WWW in the experiment. Given a set of seed URLs and a certain depth, the range can be reached by a crawler using breadth-first crawling strategy is the virtual Web. We assume that the target set *T* is the relevant set in the virtual Web, *C*(*t*) is the set of first *t* data records extracted. Therefore, we define *precision* and *recall* as follows:

$$precision = \frac{|C(t)\cap T|}{|C(t)|}\times 100\% \tag{7}$$

$$recall = \frac{|C(t)\cap T|}{|T|}\times 100\% \tag{8}$$

F-Measure, which is defined as the harmonic mean of *precision* and *recall* value, is also used to measure the performance of our method. For different specific request, according to the importance of the *precision* and *recall*, we define *F-Measure* as follows:

$$F\text{-}Measure = \frac{(\beta^2+1)\,precision\times recall}{\beta^2\times precision+recall}\times 100\% \tag{9}$$

where β is a weight for reflecting the relative importance of *precision* and *recall* value. Obviously, if $\beta>1$, then the *recall* value is more important then *precision* value. In this paper, β is assigned a constant 1.

B. Results

Before testing the efficiency and accuracy of our proposed method, we first define a threshold *lambda* (mentioned in Section III) to show how similar between the vector of the data block and the vector of domain books. *Lambda* is increased from 0.0 to 1 at interval of 0.1 during data block extraction covering hundreds of data records on domain books. The average *precision*, *recall* and *F-Measure* are shown in Fig.11 below. Fig.11 shows that with the increase of λ, *percision* increases, *recall* decreases and *F-Measure* first increases then decreases. Experimental result shows *F-Measure* reaches its peak when λ is around 0.7. So, we assign *lambda* a constant 0.7 in the remainder of the experiments.

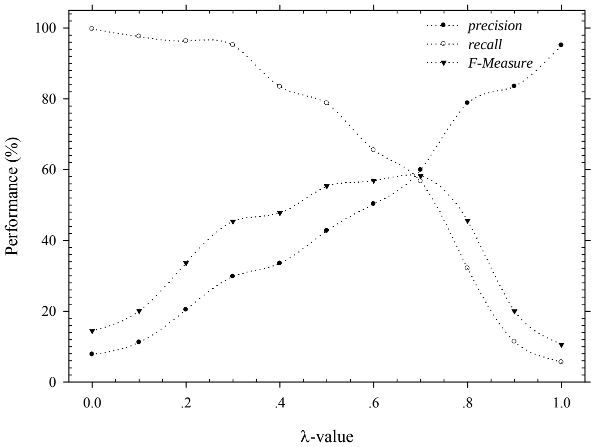


Figure 11. Dynamic plot of average *precision*, *recall* and *F-Measure* versus threshold λ. Performance is averaged across different websites.

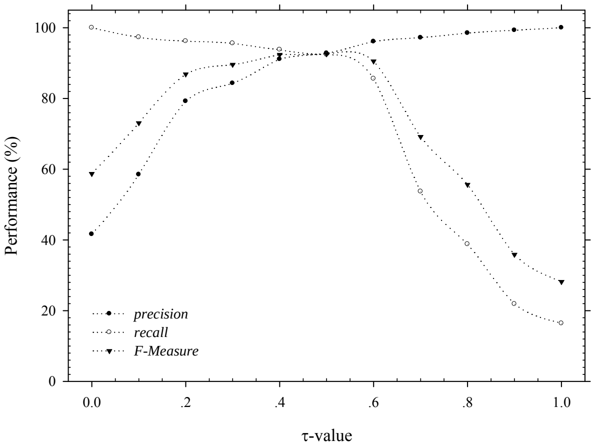


Figure 12. Dynamic plot of average *precision*, *recall* and *F-Measure* versus threshold τ. Performance is averaged across different websites.

We also analyze how different thresholds τ (the attribute similarity between ontology and Web pages) impact on post-process performance. We obtain an optimal compromise value of τ (mentioned in Section IV) by increasing it from 0.0 to 1.0 at interval of 0.1 during data record extraction covering hundreds of data records on domain books. Fig.12 shows the dynamic plot of average *precision*, *recall* and *F-Measure* versus threshold τ. According to the experimental result, *F-Measure* reach its peak when τ is around 0.5. So we assign τ a constant 0.5 in the remainder of the experiments. TABLE IV illustrates the extracting results over 4 websites in order to reflect the comprehensive of the method.

[Downloaded from www.aece.ro on Thursday, March 28, 2024 at 14:58:47 (UTC) by 3.235.139.122. Redistribution subject to AECE license or copyright.]

TABLE IV. THE EXTRACTING RESULTS OVER FOUR DIFFERENT WEBSITES.

Website	Test Number	Relevant Number	Extracted Number	Correct Extracted Number	*precision*	*recall*	*F-Measure*
www.barnesandnoble.com	150	147	143	136	0.9510	0.9252	0.937931
www.amazon.com	180	171	166	160	0.9639	0.935673	0.949555
www.chaucersbooks.com	130	123	117	112	0.9573	0.910569	0.933333
www.chapters.indigo.ca	140	132	126	118	0.9365	0.893939	0.914729

VI. CONCLUSIONS

In this paper, a novel post-processing method presented aims at using domain ontology and heuristic rules to post-process deep Web query results. The approach presented working with domain ontology as opposed to previous extracting information methods enables us to extract data records without restriction of Web page structures and database designing styles. Besides, semantic meanings and relations are added which makes extracting data information more accurate. The approach also defines two models (*BIM* and *RSEM*) to extract potential data block based on node similarity and calculate similarity based on *VSM*. Some irrelative parts such as ads, navigation and flash, are get rid of in the process of reducing noisy. Four heuristic rules are proposed to annotate extracting results so that the results can match the attributes in domain ontology. In addition, basic information is extracted to simplify final results. Finally, we propose a ranking algorithm to present users an ordered list of data records. The experimental results verify that our proposed method could extract data information efficiently and accurately. Consequently, our post-processing method can be designed to extract all possible information of the deep Web across multiple topics.

REFERENCES

[1] I.A. Letia and A. Marginean, “Client provider collaboration for service bundling,” *Advances in Electrical and Computer Engineering*, Vol. 8, no. 1, pp. 36-43, 2008.

[2] C.H. Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan, “A survey of Web information extraction systems,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1411-1428, 2006.

[3] R. Grishman and B. Sundheim, “Message understanding conference-6: a brief history,” *In Proc. Of the16th Int’l Conf. on Computational Linguistics (COLING -96)*, August 1996.

[4] D.G. Gregg and S. Walczak, “Exploiting the information Web,” *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Review*, vol. 37, no. 1, 2007.

[5] H. He, W.Y. Meng, and Y.Y. Lu, “Towards deeper understanding of the search interfaces of the deep web,” *Word Wide Web Journal*, vol. 10, no. 2, pp. 133-155, 2007.

[6] H. He, W. Meng, C.T. Yu, and Z. Wu. “WISE-integrator: an automatic integrator of web search interfaces for e-commerce,” *In Proceedings of the 29th International Conference on Very Large Data Bases(VLDB), Berlin*, pp: 357-368, 2003.

[7] H. He, W.Y. Meng. C. Yu, and Z.H. Wu, “Constructing interface schemas for search interfaces of web databases,” *In Proceedings of WISE*, pp: 29-42, 2005.

[8] X. Peng and Z. Huang, “Enabling semantic queries against the spatial database,” *Advances in Electrical and Engineering*, Vol. 12, no. 1, pp. 45-50, 2012.

[9] R.B. Doorenbos, O. Etzioni, and D. Weld, “A scalable comparison shopping agent for the World Wide Web, ” *Proc of the First International Conference on Autonomous Agents*, Marina del Rey, CA, pp. 39-48, 1997.

[10] L. Gravano, P.G. Ipeirotis, and M. Sahami,“QProbe: a system for automatic classification of hidden-Web databases,”*ACM Transactions on Information Systems*, vol. 21, no. 1, pp. 1-41, 2003.

[11] F. Ashraf, T. Ozyer, and R. Alhajj,“ Employing clustering techniques for automatic information extraction from HTML documents,” *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Review*, vol. 38, no. 5, 2008.

[12] B. Liu and Y.H. Zhai, “NET-a system for extracting web data from flat and nested data records,” *Web Information Systems Engineering-WISE 2005*, *Lecture Notes in Computer Science*, vol.3806, pp. 487-495, 2005.

[13] J.L. Hong, “Data extraction for deep web using WordNet,” *IEEE Transactions on Systems Man and Cybernetics Part C-Appliacations and Reviews*, vol.41, no.6, pp. 854-868, 2011.

[14] N. Marian and S. Top, “Integration of simulink models with component-based software model,” *Advances in Electrical and Computer Engineering*, Vol. 8, no. 2, pp. 3-10, 2008.

[15] L. Stanescu and D. Burdescu, “Information structuring and retrieval with topic maps for medical e-learning,” *Advances in Electrical and Computer Engineering*, Vol. 9, no. 3, pp. 27-33, 2009.

[16] S. Slderland, “Learning information extraction rules for semi-structured and free text,” *Machine Learning*, vol. 34, nos. 1-3, pp. 233-272, 1999.

[17] V. Crescenzi, G. Mecca, and P. Merialdo, “RoadRunner: towards automatic data extraction form large Web site,” *Proceeding of the 27th. International Conference on Very Large Data Bases*, Roma, pp. 109-118, 2001.

[18] D. Cai, S.P. Yu, J.R. Wen, and W.Y. Ma, “VIPS: a vision-based page segmentation algorithm,” *Microsoft Technical Report*, MSR-TR 2003-79.

[19] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu, “ Fully automatic wrapper generation for search engines,” *In Proceedings of the 14th World Wide Web Conference*, pp. 66-75, 2005.

[20] Z.P. Wang, Y.G. Zhang, J.F. Zhang, and J. Ma, “Recent research process in fault analysis of complex electric power systems,” *Advances in Electrical and Computer Engineering*, Vol. 10, no. 1, pp. 28-33, 2010.

[21] B. Liu, R. Grossman, and Y. Zhai, “Mining data records in web page,” *In SIGKDD’03*, 2003.

[22] W.C. Bruce, M. Donald, and S. Trevor, “Search engines: information retrieval in practice,” Addison Wesley, 2009.

[23] M. Horridge, B. Parsia, and U. Sattler, “Explanation of OWL entailments in protege4,” *In Proceedings of International Semantic Web Conference*, 2008.

[24] A. Bilke and F. Naumann, “Schema matching using duplicates, ” *In Proceedings of the 21st IEEE International Conference on Date Engineering*, pp. 69-80, 2005.

[25] Y.Y, Lu, W.Y. Meng, L.C. Shu, C. Yu, and K.L. Liu, “Evaluation of result merging strategies for metasearch engines,” *6th International Conference on Web Information Systems Engineering(WISE05)*, New York City, pp. 53-66, November 2005.