

Data Extraction and Annotation Based on Domain-specific Ontology Evolution for Deep Web

Kerui Chen^{1,2}, Wanli Zuo^{*1}, Fengling He¹, Yongheng Chen¹ and Ying Wang¹

¹ College of Computer Science and Technology, Jilin University, Changchun 130012, China

Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education

² College of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou 450002, China

*Corresponding Author: Wanli Zuo wanli@jlu.edu.cn

Abstract. Deep web respond to a user query result records encoded in HTML files. Data extraction and data annotation, which are important for many applications, extracts and annotates the record from the HTML pages. We proposed an domain-specific ontology based data extraction and annotation technique; we first construct mini-ontology for specific domain according to information of query interface and query result pages; then, use constructed mini-ontology for identifying data areas and mapping data annotations in data extraction; in order to adapt to new sample set, mini-ontology will evolve dynamically based on data extraction and data annotation. Experimental results demonstrate that this method has higher precision and recall in data extraction and data annotation.

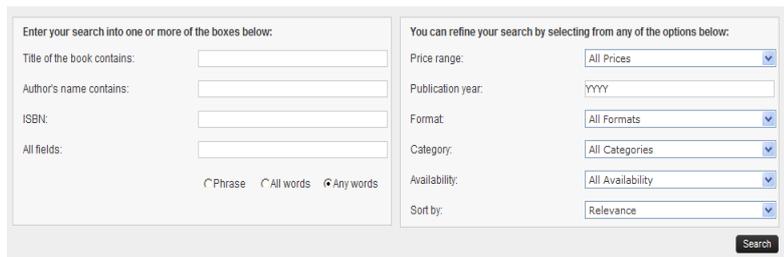
Keywords: Deep Web, Data Extraction, Data Annotation, Domain Ontology, Ontology Evolution.

1. Introduction

Currently, more and more researchers start focusing on how to manage data information hided in back-end database more effectively. As a result, according to the distribution of Web and storage depth of information, Web has been classified as "Surface Web" and "Deep Web" (Hidden Web/Invisible Web). Surface Web represents internet resources that are linked by hyperlink, such as picture, file, and static web page, and they usually could be accessed by using these hyperlinks; on the other side, Deep Web is constructed by back-end databases, and their contents are stored in relational databases of back-end web sites; unlike Surface Web, there are no hyperlinks to these web sites, it rather dynamically produces web sites contained query result records by back-end server based on query conditions.



(a) Simple of Query interface



(b) Advanced Query interface



(c) Query result display

Fig. 1. The sample of Query interface and Query result

From Figure 1, we could figure out that (a) and (b) represents two query interfaces. After inputs keyword “c program”, the server will get query result (c). The main task in this paper is to annotate each data item in (c), and to integrate results returned by various data sources into one table.

Through automatic data extraction in Deep Web, with data integration by data annotation, it would be able to provide better service to various commercial web sites, such as the seller or agency of internet commercial information; in addition, it also helps portals to provide more professional and personalized information search service.

2. Related work

In recent years, there are more and more web data extraction tools coming out, and they could be classified by functions and features into categories as follows: Languages for Wrapper Development [1-3], HTML-aware Tools [4-5], NLP based(natural language processing)[6-7], wrapper induction based[8-9], data modeling based[10-11], visual information based[12-13], and ontology based web extraction method[14].

The research on web data extraction is comparatively mature, and there are many data extraction methods available from theory to practice; on the other side, the research on data annotation is still in the infant stage both domestically and internationally. There are mainly three types of semantic annotation method targeted at data: mode based system [15-16], machine learning based [17-18], and ontology based method [19-21]. For data annotation, ontology based method is the main streaming. There are more and more researchers start to accept practicability of using ontology in data annotation, and actually make some achievements.

Multiple annotation tools mentioned above adopts heuristic rules for annotating. If only a specific Deep Web database needs to be annotated, it will be possible to use machine learning algorithm to train in sample training set; once semantic relationship between data was obtained, it would dig out a series of rule sets and apply them in annotating new web sites. Although this method is only available in specific Deep Web pages, it does not work for other Deep Web pages in the same domain; therefore, simply use machine learning method cannot suit a mass of isomorous Deep Web pages.

While some annotation tools did use ontology, many problems still existed even they suited for annotation of multiple domains. For example, majority of annotation tools are targeting at pure text, so their ontology definitions are relatively complex, less efficient, and not adapting to Deep Web's structural data features. Moreover, as most defined ontology are static, precision and adaptability needs to be improved.

The organization structure of this paper is listed as below: the third section discussed construction of mini-ontology; then, the following section detailedly introduced how to use ontology information to identify data records area; based on predefined division and alignment rule, data in data records area would be divided and aligned; in this way, each data record would be extracted out; the fifth section mainly focused on using ontology to annotate extracted data records; finally, in sixth section, we suggested a evolution frame to examine how mini-ontology evolved with data extraction and annotation; The last two sections evaluate experiments and provide a future outlook.

3. Mini-ontology generation

3.1. Definition

PVAs(Programmer Viewpoint Attributes): attributes extracted from query interface in the view of web programming. These PVAs extracted from HTML labels were similar to values of following labels: <label>, <input>, <option>, <select> and so on.

UVAs (User Viewpoint Attributes): attributes extracted from query interface in the view of web users. Normally, values of UVAs follow the label closely.

3.2. Attribute recognition

The extraction of query interface's attributes took example form Automatic Attribute Extraction method proposed by YOO JUNG AN et.al [22], there were two steps for extracting attributes from query interfaces:

(1)To acquire PVAs from source code of query interfaces;

The extraction process of PVAs is executed in the order as follows:

Step 1: to extract all string sets SS from Html document's tags, and to record keysets between label <label> and </label>, as well as label <Select> and </Select>, in KW; meanwhile, duplicate ones will be removed;

Step 2: to traverse each string in SS for identifying whether special symbols like “.”, “/”, “{”, “}”, “#”, “\$”, “&”, “*”, “>”, “+”, “\”, “=”, “?”, “<”, “[”, “]”, “@”, “_” were included; if so, the string would be divided into two sub-strings by using symbols above as dividing line;

Step 3: to traverse each string in SS for identifying whether capitalized letters were included; and then use capitalized letters as dividing line to divide string into two sub-strings;

Step 4: to traverse each string in SS for identifying whether key words in set KW were included; if so, this query key word would be used as boundary for dividing string into two sub-strings;

Step 5: to record NSS, the number of times that string SS_i has been divided;

Step 6: to recombine sub-string SS_i, divided from each string; while only neighboring sub-strings could be combined, the number starts from 2 until its value reaches NSS_i, and combined strings would be stored in SS_i.

Step 7: To recheck string set SS and to delete duplicated strings;

Data Extraction and Annotation based on Domain-specific Ontology Evolution for Deep Web

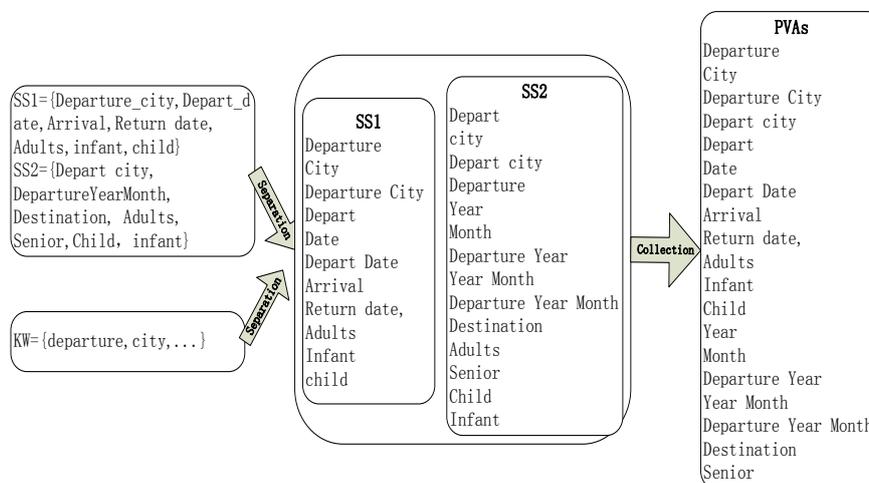


Fig. 2. Sample process of acquiring PVAs

Shown as fig.2, there were two original string sets SS1 and SS2 extracted from query interface's HTML labels, and key word set KW extracted between <label> and </label>, and <Select> and </Select>.

We first divide SS1; since "Departure city" and "Depart date" both contain special symbol "_", they should be divided into four strings: "Departure", "city", "Depart" and "date". Right after the division, the combining process will start. For instances, "Departure" and "city" could be combined as "Departure city", while "Depart" and "date" could be combined as "Depart date"; the final output set is showing as SS1 in the second circle.

Then, we start dividing SS2. As "Departure city" contains "city" from key word set KW, it needs to be divided. Similarly, "DepartureYearMonth" would be divided into "Departure", "Year", and "Month" as it contains capitalized letter "Y" and "M". After division, combining process will start again. "Departure", "Year" and "Month" could be combined as "Departure Year", "Year Month" and "DepartureYearMonth"; the final output set is showing as SS2 in the second circle.

Finally, we combine SS1 and SS2, and pick out duplicated string once checked out, such as "Departure", "Depart", "city" and so on; after that, PVAs set would be obtained.

(2)To acquire UVAs from texts of query interfaces;

The steps of extracting UVAs from query interfaces are listed below:

Step 1: to traverse HTML source codes for searching start tag <Option> and end tag </Option>; then, to remove all free texts between these two tags.

Step 2: to traverse HTML source code again, and to save free texts between any two labels in set UVAs in the form of strings.

Step 3: to remove duplicated texts in UVAs.

Step 4: to traverse all strings in set UVAs.

When there is a string that contains special symbols, the special symbol will be used as dividing line to divide that string into two sub-strings, which will be stored in set UVAs instead of previous string.

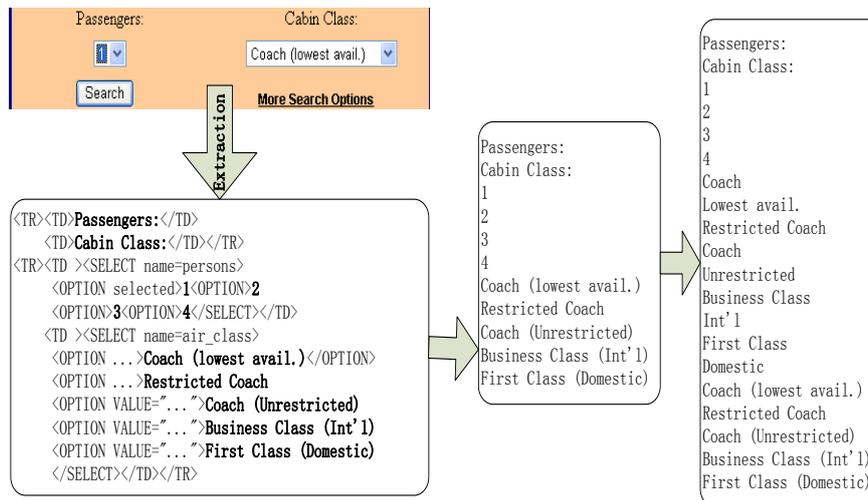


Fig. 3. Sample process of acquiring UVAs

Shown as fig .3, UVAs is the instance extracted from labels, such as the free texts between <TD> and </TD>: “Passengers:” and “Cabin Class:”, and free texts between <Option> and </Option>: “1”, “2”, “Coach (lowest avail.)”, “Restricted Coach” and so on. The instance value needs to be judged whether it contains special symbols in order to divide. Since “Coach (lowest avail.)” contains “(”, it will be divided into “Coach” and “lowest avail.”. Then, the set UVAs will be obtained.

3.3. Mini-ontology generation

Not only does construction information of Mini-ontology include attribute information of query interface, but it also contains abundant instance information in query result pages. The extraction of query result data instance and construction of ontology adopts method in [23].

4. Automatic data extraction

Currently, data extraction is facing challenges as below:

- (a) Data have different layouts and patterns in web sites;
- (b) It's more difficult to dig right relevancy between labels as a label in the table was constructed by cells with multiple levels;

- (c) Data items use different expression forms;
- (d) There were isolated information and those may cause ambiguity in the table;
- (e) It's hard to identify and extract when single data record existed in some web sites.

In order to solve problems mentioned above, this paper suggests an ontology based extraction method, which had been confirmed in experiment for its effectiveness of extracting data in web sites.

In fig .4, the whole process of data extraction has been demonstrated; the first step is to convert query result pages to DOM trees; then, to identify interested data areas by using ontology's label classifier and instance classifier; the third step is to divide data records based on the predefined heuristic rules; finally, to align extracted data records.

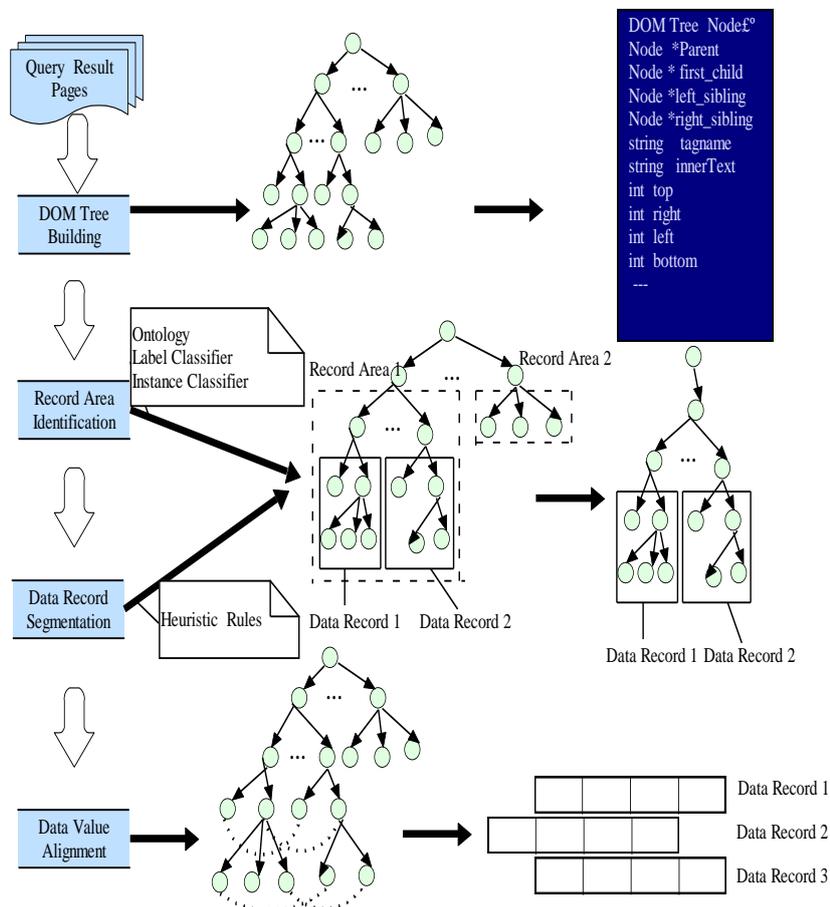


Fig. 4. Data extraction procedure

4.1. DOM tree building

Usually, a tag contains a pair of switch symbols, such as “<” and “>”, and a tag also contains a set of tag attributes. This paper designs tags in two types: start tag and end tag. As these tags are paired, the only difference is the extra symbol “/” in the beginning of end tag. For example, in figure 3.2, <table> is the start tag while </table> is the end tag. In the process, there were two types of tags that will not be considered: the first is the tag begins with “<!”, while another one is the end tag that has no associated start tag.

For extraction of query result pages instances, we first convert result pages to DOM trees. The construction method used in this paper is to use tags with visual aids, which can help deduce structural relationship between tags and construct stronger DOM tree.

The construction steps were shown as follows:

- (1) To use browser’s rendering engine to find out four boundary values of each element in HTML;
- (2) To check and construct from first tag in the source code of documents.

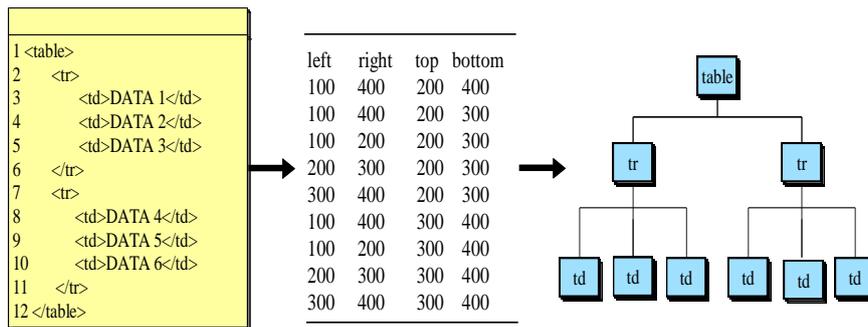


Fig. 5. HTML encoding sample, boundary coordinate and tag tree structure

Fig.5 demonstrates the construction process of DOM tree; first, the analysis chart provides source code for a table with 2 rows and 3 columns; then, it uses browser’s rendering engine to acquire boundary values of tag <table>, <tr>, and <td>; for example, the four boundary values of tag <table> are left: 100, right: 400, top: 200, and bottom: 400; finally, the tree structure of tag will be constructed according to four boundary values. Generally, browser’s rendering engine has a very high fault tolerance capability, and therefore the encoding mistakes existed in source code could be identified correctly and those constructed boundary coordinates have higher accuracy rate.

4.2. Data area identification

Although we can obtain multiple data areas, there were only one or several data areas we are interested in; hence, we need to further dig out interested data areas.

In the process of mining interested data areas, this paper synthetically utilizes methods of ontology and heuristic rules.

Heuristic rules are based on the observations as below:

(a) A group of data records containing similar target sets usually appear in the neighboring areas of web sites; in addition, they also have similar HTML tags.

(b) A data area's data records list is made up by multiple subtrees of the same father node. In other words, these data records could be considered as multiple subtrees while they have the same father node.

(c) Learning through observation, abundant data records areas usually locate at the centre of web site.

(d) If the depth of leaf node in DOM tree was too low, the node would be determined to be useless.

(e) If the data volume of node's neighboring area was too low, this node would also be useless.

The ontology used for mining data areas is based on two observations as below [30]:

(a) Data areas contain plentiful ontology information.

(b) Attributes and instances contained in ontology are usually located closely in data areas.

(1) Ontology instance relevancy:

For a given ontology O , instance set $D = \{d_1, d_2, \dots, d_n\}$, which comes from data records; for any original string d_i , its weight was represented as w_i ; if $d_i = A_i.N$ or $d_i = A_i.FA_i$, which means d_i is the name or alias of ontology O 's attribute A_i , the value of w_i will be the probability that attribute A_i appears. If d_i is the value of multiple attributes, then the value of w_i will equals to the highest probability for attribute's appearance.

In order to calculate relevancy between D and O , the following formula will be used:

$$Corr(D, O) = \frac{\sum_{i=1}^n w_i}{\sqrt{mn}} \quad (1)$$

In the formula, m represents the number of attributes contained in ontology O , while n represents the number of data items in the instance set.

(2) Example of data extraction algorithm

```
SearchDataRegion(R, O)
{ T ← Ri ∈ R;
  While(T is not leaf node)
    begin Ti ← one child node of T has the largest
correlation with O;
      if (Corr(T, O) < Corr(Ti, O)) T ← Ti
      else break
    End
  While(1)
  Begin Tl ← the left sibling of T
    If (Corr(T, O) < Corr(Tl+T, O)) T ← Tl+T
    else break
  End
  While(1)
  Begin Tr ← the right sibling of T
    If (Corr(T, O) < Corr(Tr+T, O)) T ← Tr+T
    else break
  End
  Return T
}
```

The algorithm mentioned above is mainly used for searching subtree that has maximum relevancy with given ontology, and hence concludes that the subtree actually is the data area we are looking for. *DT* is the set of data areas divided by web sites; only one subtree of the set will be picked up each time; moreover, starts from root node of that subtree, by following the order of top-down, left to right for nodes in the same level, to identify an subtree with highest relevancy.

4.3. Data record segmentation

If a data area only contains one data record, it will not be necessary to divide records; to the contrary, if it contains multiple data records, we will need to seek for an algorithm for executing division of data records.

By observing structural characteristics and encoding characteristics of data records in data areas, we propose a comprehensive division algorithm, which can utilize characteristics of tags effectively.

(1) Statistics the max number label

For candidate tags in the nodes of sub tree that has greatest Fan out of they would be ranked in an inverted order according to the number of appearance; as a final product, a annotation serial will be obtained.

(2) Identify division tag

For the layout of documents contained multiple data records, we tend to use few division tags to divide those records, such as tags: <hr>, ,
 and so on. Based on this observation, we can track these division tags for inspecting its frequency of being used.

(3)Standard deviation

We calculate standard deviation between each candidate tag.

(4)Repeat mode of tags

In a data record area T=ABABABA, as A and B represent two different types of tag structure, we can find two repeat mode: AB and BA. If there is only one repeat mode, then we will be able to determine the repeat mode of data record area; otherwise, we need to choose one. At this time, we can use visual aid as gap between two data records is usually greater than the one between data items in the same data record; by applying this limit, we can delete useless repeat modes.

Based on the four heuristic rules discussed above, we can divide data records in data records areas and align data records.

4.4. Data value alignment

| | | |
|---|--|--|
|  | Corrective Reading Program Undefined 03 Nov 1998 Available to Order. Publisher stock level unknown. Typical order time is 1-3 weeks, but varies by publisher. 0 in stock at Charing Cross Road. | List Price: £19.99 Buy Now ▶ More Info ▶ |
|  | C# Program & Progress by Shanbedi, Mahmood Paperback 15 Aug 2005 Available to Order. Publisher stock level unknown. Typical order time is 1-3 weeks, but varies by publisher. 0 in stock at Charing Cross Road. | List Price: £19.95 Buy Now ▶ More Info ▶ |
|  | C How to Program C Student Solutions Manual by Deitel, Harvey M. Mixed media product 19 Dec 2003 Not available | List Price: £19.94 More Info ▶ |

Fig. 6. Query result instance sample

Fig.6 shows a real data record sample, which can be divided into 3 data records. Table 1 demonstrates result of the three data records' alignment; from observation, we learned that numbers of nodes contained in these three data records are different. Unless the formwork of Deep Web was produced

strictly, the case mentioned above would often appear. Hence, after the data division, we still need to align divided data records.

Table 1. Fig. 6 alignment of data records

| | | | | |
|---|-----------------------|---------------------|-------------|------------------------|
| Corrective Reading program | | Undefined | 03 Nov 1998 | Available to order.... |
| C# Program & Progress | by Shanbedi, Mathmood | Paperback | 15 Aug 2005 | Available to order.... |
| C How to Program Student Solutions Manual | By Deitel, Harvey M | Mixed media product | 19 Dec 2003 | Not available |

For alignment of data records in this paper, the partial tree alignment algorithm [24] has been used for aligning multiple data records produced by same data source. The main idea of this algorithm is to create an incremental seed tree for aligning multiple trees. If we consider a data record as a seed tree, a data area will have subtrees same with the number of data records contained.

5. Automatic data annotation

After the extraction of Deep Web and identification of duplicated records mentioned above, we can get an instance set $I = \{i_1, i_2, \dots, i_n\}$, $\forall i_m \in I$, $i_m = (l_m, d_m)$; among it, l_m represents a tag, whose value could be null; while d_m represents an instance value, $d_m \in D$. Since d_m appears in various web sites, it's possible to have different l_m ; the annotation of data is to annotate a unified tag for d_m , which would help integrate data.

There were two types of treatments for l_m : when l_m is not null, we will adopt method suggested in [25]; when it's null, which means the attribute value is original and has no page tags, there will be two methods of annotating; in addition, the results of annotation will be stored in l_m and mapping relationship between l_m and ontology O will also be recorded eventually.

The first one adopts "Query Reset Strategy" proposed in reference [21], which was created based on observations as below: the more appropriate query conditions has been chosen, the more query result information would be returned by Deep Web background server.

The second one is based on the K-beam search algorithm of KBFS. Not only does this algorithm utilize prediction method of prediction model based on maximum information entropy models, but it also takes advantage of KBFS search algorithm's ability of seeking for optimal path. The construction steps are listed as below:

Based on maximum information entropy models, we constructed a prediction model for annotation prediction of d_m in instance set $I = \{i_1, i_2, \dots, i_n\}$, $\forall i_m \in I$, $i_m = (I_m, d_m)$. The construction of this model used maximum possibility attribute model [26] as reference.

For a given instance value d_m , h is the characteristic of d_m when it appeared previously, the possibility that each attribute A_i in attribute set A contained in ontology O will be annotated by d_m is determined by prediction model as follows:

$$p(h, A_i) = \pi \prod_{j=1}^k \lambda_j^{f(h, A_i)} \quad (2)$$

In the formula, π is a constant, $f(h, A_i)$ is the characteristic function, and its value is either 0 or 1, which is also the weight λ_j of each characteristic in characteristic set. The possibility that instance value d_m will be annotated as attribute A_m is calculated as below:

$$p(A_m | h) = \frac{p(h, A_m)}{\sum_{A_i \in A} p(h, A_i)} \quad (3)$$

While $\sum_{A_i \in A} p(h, A_i)$ represents the sum of probability for all attributes.

Hence, for an original data record's data item set $\{d_1, d_2, \dots, d_m\}$, the conditional probability that annotated attribute tag order is $\{A_1, A_2, \dots, A_m\}$ will be:

$$p(A_1, A_2, \dots, A_m | d_1, d_2, \dots, d_m) = \prod_{i=1}^m p(A_i | h_i) \quad (4)$$

By using calculation of prediction models mentioned above, we can get multiple prediction values; once we construct a tree based on those values, it's time to use KBFS search algorithm[27] to seek for a optimal path in prediction values' constructed tree.

6. Ontology evolution

Ontology evolution could be considered as improving ontology for adjusting to new sample sets or user requests without violating ontology's compatibility principle.

(1) Frame of ontology evolution

The frame of ontology evolution in this paper was inspired by reference [28], and it mainly has four stages as below:

(a) Capturing change information

The procedure of ontology evolution starts from capturing change information, which has two types: change based on structure, and change based on data; moreover, change based on structure requires modifying the structure of ontology, while change based on data only needs to modify associated data.

(b) Expressing change information

In order to handle captured change information, there is a further need to identify and express them in proper form; in other words, various types of change information all need related expressing method.

(c) Semantic change

Before the execution of ontology evolution, it's necessary to check whether there is a semantic change existed. The dependency between attributes in the ontology needs to be checked carefully, as they may cause ambiguous problems.

(d) Execution of ontology evolution

In this stage, the main work is to send a request of modifying ontology; once the modification has been executed, it will be recorded in order to achieve operation's reversibility.

Next, there will be detailed explanations for each stage.

(2) Capturing information change stage

Each time, after the extraction and annotation, the system will return some information to ontology evolution module. This information, including extracted instances that have not been annotated yet, will be annotated as one attribute of ontology; however, that ontology does not contain that instance value, or instances with unsure annotation; in this case, ontology evolution module needs to classifier the information based on actual situation, in order to execute expressing change information in next stage.

(3) Expressing change information stage

In this stage, classified change information would be expressed respectively. If instance that has not been annotated requires to construct a new attribute model manually for it, or ontology does not contain instance value which has been annotated as an attribute of ontology, there will be a need to create an relationship between attributes of instance and annotation; in case there are instances whose annotation were uncertain, the uncertain attribute tags should be recorded.

(4) Checking semantic change stage

Before execution of ontology evolution, there will be a semantic check for ontology operation that will be executed; for example, to check whether new

attribute model has synonymous or dependence with current attributes of ontology.

(5) Ontology evolution execution stage

Based on the special tags made by ontology during annotation, ontology evolution needs to follow rules as below:

Rule 1: when deals with data of mapping relationship between N_m and l_m , the location of ontology attribute A_m will be searched, and d_m , in the instance $i_m = (l_m, d_m)$, will be added to V_m .

Rule 2: when deals with data of mapping relationship between A_m and l_m , the location of ontology attribute A_m will be searched, and l_m , in the instance $i_m = (l_m, d_m)$, will be added to FA_m .

Rule 3: when deals with data of mapping relationship between l_m and ontology O , then a new attribute A_x will be created according to the instance $i_m = (l_m, d_m)$, and l_m will be the value of name entry N in attribute A_x tuple while d_m will be the value of attribute entry V .

As new ontology will be created after the evolution each time, multiple ontology versions will be created with the multiple evolution; hence, it's necessary to manage versions in order to prohibit losing data and to maintain ontology's consistence effectively.

7. Empirical evaluations

Test sample selected three domains from UIUC concentrating storehouse[29]: Automobile, Airfares, and Books; in each domain, 10 query interfaces has been chosen, and been provide query conditions randomly typed; then, those returned query result pages would be collected manually; in case that one query condition returned multiple query results, only first page would be selected while others would be ignored.

We manually classify collected web sites in accordance with three categories listed as below:

- (a) Web site contained multiple data records (MRP)
- (b) Web site contained one data records (SRP)
- (c) Web site did not contain data records (ERP)

7.1. DataSet

There were totally 60 web sites collected from three domains: Automobile, Airfares, and Books, and the statistics of data records distribution in these web sites were shown as table 2. Since the typed query conditions were comparatively appropriate, there were less SRP and ERP in the collection. Moreover, there were more data records returned from Airfares domain, while

less from Automobile domain; this phenomenon was actually determined by features of domain.

Table 2. Statistics of data set

| Domain | Num(Web pages) | Num(MRP) | Num(SRP) | Num(ERP) |
|------------|----------------|----------|----------|----------|
| Automobile | 20 | 120 | 37 | 43 |
| Airfares | 20 | 175 | 9 | 16 |
| Books | 20 | 162 | 11 | 27 |
| Total | 60 | 457 | 57 | 86 |

7.2. Performance of ontology based data area identification algorithm

The performance of ontology based data area identification was shown as table 3, Airfares domain reached 100% for identifying MRP, while Books domain has lowest score, which was only 89.4%. From this, we can conclude that ontology has highest identification efficiency in Airfares domain. Books domain had lowest scores for MRP, SRP, and ERP, which could be explained by the relatively complex query interfaces in this domain.

Table 3. Performance of ontology based data area identification algorithm

| Domain | MRP | SRP | ERP |
|------------|-------|-------|-------|
| Automobile | 91.2% | 89.7% | 85.9% |
| Airfares | 100% | 96.5% | 94.2% |
| Books | 89.4% | 84.4% | 83.1% |
| Average | 93.5% | 90.2% | 87.7% |

The main idea of ontology based data area extraction method is that areas contained massive ontology information is more likely where query results located. As Books domain contained a lot of books recommendation information, which usually have complete description, they could disturb real query result records very easily, and this disturbance would be more severe when there were only a few query result records; therefore, we can find that the precision of Books domain for ERP was only 83.1% in the table.

7.3. Performance of data extraction and annotation after ontology evolution

In newly collected test sample set, the comparison for performance of ontology evolution based data extraction was shown as Table 4. We can easily find that newly evolved ontology performed better than original ontology in each domain, especially for Books domain, whose recall has been

increased by 0.9%, although Automatic domain did not increase drastically. These three domains increased 0.17% for their precisions, and 0.7% for recall. In this way, experimental results proved that improved ontology performed better in data extraction.

Table 4. performance comparison of data extraction in new training set

| Domain | Mini-ontology | | Ontology after evolution | |
|------------|---------------|--------|--------------------------|--------|
| | Precision | Recall | Precision | Recall |
| Automobile | 80.2% | 78.1% | 80.3% | 78.5% |
| Airfares | 92.9% | 92.8% | 93.0% | 93.6% |
| Books | 86.8% | 83.9% | 87.1% | 84.8% |
| Average | 86.63% | 84.93% | 86.80% | 85.63% |

Table 5. performance comparison of data annotation in new training set

| Domain | Mini-ontology | | Ontology after evolution | |
|------------|---------------|--------|--------------------------|--------|
| | Precision | Recall | Precision | Recall |
| Automobile | 85.0% | 84.7% | 85.2% | 85.3% |
| Airfares | 97.1% | 95.6% | 97.1% | 95.7% |
| Books | 86.3% | 84.9% | 86.7% | 85.7% |
| Average | 89.47% | 88.40% | 89.67% | 88.90% |

In newly collected test sample set, the comparison for performance of ontology evolution based data annotation was shown as Table 5. Just as expected, newly evolved ontology performed better than original ontology and there were increase in each domain with varying degrees. The increase in Airfares domain was not drastic as there were few attributes in this domain and relationships between attributes were also quite simple. Hence, the evolution degree of ontology in Airfares domain was not obvious, and this had little effect on annotation. In Books domain, the increase is relatively high; actually, precision and recall was increased by 0.4% and 0.8%, respectively. Experimental results demonstrated that newly evolved ontology performed better in annotating data, compared to original ontology.

8. Conclusion and future work

This paper takes full advantage of ontology's semantic information; in the process of identifying query result data areas, with the guidance of ontology, the identification will be accurate; in addition, it also solved dependence problems on web site structures existed in traditional template based method. This paper also systemically and deeply investigates design of ontology module adjusting to structural web sites, automatic construction of domain

ontology, extraction of data records in query result pages, annotation of data records, as well as the evolution mechanism of ontology.

Regarding research works discussed in this paper, there are still a lot of things need to be improved. For example, the construction process of ontology could be improved, in order to solve some ambiguous problems caused by attributes matching; furthermore, an ontology evolution evaluation mechanism also needs to be set up for controlling the increasing ontology data volume, and thereby seek for a balance point between efficiency and accuracy.

Acknowledgment. This work is supported by the National Natural Science Foundation of China under Grant No.60973040; the National Natural Science Foundation of China under Grant No.60903098; the Science and Technology Development Program of Jilin Province of China under Grant No. 20070533; the Specialized Research Foundation for the Doctoral Program of Higher Education of China under Grant No.200801830021; the Basic Scientific Research Foundation for the Interdisciplinary Research and Innovation Project of Jilin University under Grant No.450060445161; the Basic Scientific Research Foundation for Young Teachers Innovation Project of Jilin University under Grant No.450060441075.

References

1. Crescenzi V, Mecca G.: Grammars have exceptions. *Information Systems*, Vol. 23, No. 8, 539-565. (1998)
2. Hammer J, Mchugh J, Garcia-Molina H.: Semistructured data: The TSIMMIS experience. In *proceedings of the first east-european symposium on advances in databases and information systems*, 1-8. (1997)
3. Arocena G.O, Mendelzon A.O.: WebOQL: Restructuring documents, databases and webs. In *proceedings of the 14th international conference on data engineering*, 24-33. (1998)
4. Crescenzi V, Mecca G, Merialdo P.: Roadrunner: Towards automatic data extraction from large Web sites. In *proceedings of the 26th International Conference on Very Large Databases*, 109-118. (2001)
5. Baumgartner R, Flesca S, Gottlob G.: Visual Web information extraction with Lixto. In *proceedings of the 26th International Conference on Very Large Databases*, 119-128. (2001)
6. Califf M.E, Mooney R.J.: Relational Learning of Pattern-Match Rules for Information Extraction. In *proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence*, 328-334. (1999)
7. Freitag D.: Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, Vol. 39, No.2-3, 169-202. (2000)
8. Kushmerick N.: Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence Journal*, Vol. 118, No. 1-2, 15-68. (2000)

9. Muslea I, Minton S, Knoblock C.: Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, Vol. 4, No. 1-2, 93-114. (2001)
10. Adelberg B.: NoDoSE-A tool for semi-automatically extracting structured and semistructured data from text documents. In proceeding of the ACM SIGMOD international Conference on Management of Data, 283-294. (1998)
11. Laender A H F, Ribeiro-Neto B, Da Silva A S.: DEByE-Data Extraction By Example. *Data and Knowledge Engineering*, Vol. 40, No. 2, 121-154. (2002)
12. Deng Cai, Yu Shipeng, Wen Jinrong, et al.: VIPS: A Vision-Based PageSegmentation Algorithm. Microsoft Technical Report, MSR-TR-203-79. (2003)
13. Liu W, Meng X, Meng W.: ViDE: A Vision-Based Approach for Deep Web Data Extraction. *Knowledge and Data Engineering*, Vol. 22, No. 3, 447-460. (2010)
14. Data Extraction Research Group. <http://www.deg.byu.edu/>.
15. S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K.S. Mccurley, S. Rajagopalan, A. Tomkins.: A case for automated large-scale semantic annotation. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 1, No. 1, 115-132. (2003)
16. B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, A. Kirilov.: KIM - a semantic platform for information extraction and retrieval. *Natural Language Engineering*, Vol. 10, No. 3-4, 375-392. (2004)
17. L. Arlotta, V. Crescenzi, G. Mecca, P. Merialdo.: Automatic Annotation of Data Extracted from Large Web Sites. In proceedings of Sixth International Workshop on the Web and Databases (WebDB 2003), 7-12. (2003)
18. J. Wang F.H. Lochovsky.: Data Extraction and Label Assignment for Web Databases. In proceedings of the 12th international conference on World Wide Web, New York, USA, 187-196. (2003)
19. Ciravegna F, Dingli A.: User-System Cooperation in document Annotation based on Information Extraction. In proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, Siguenza, Spain, 65-80. (2002)
20. Yihong Ding, David W. Embley.: Using Data-Extraction Ontologies to Foster Automating Semantic Annotation. In proceedings of the 22nd International Conference on Data Engineering Workshops. (2006)
21. Liu Yuan, Zhanhuai Li, Shiliang Chen.: Ontology-Based Annotation for Deep Web Data. *Journal of software*, Vol. 19, No. 2, 237-245. (2008)
22. Yoo Jung An, James Geller, Yi-Ta Wu, Soon Ae Chun.: Semantic deep web: automatic attribute extraction from the deep web data sources. *Symposium on Applied Computing*, 1667-1672. (2007)
23. Chen Kerui, Zuo Wanli, Zhang Fan, He Fengling, Peng Tao.: Automatic generation of domain-specific ontology from deep web. *Journal of Information and Computational Science*, Vol. 7, No. 2, 519-525. (2010)
24. Yanghong Zhai, Bing Liu.: Web Data Extraction Based on Partial Tree Alignment. In proceedings of World Wide Web Conference 2005, 76-85. (2005)
25. 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*, unpublished. (2011)
26. Adwait Ratnaparkhi.: A maximum entropy model for part-of-speech tagging. In proceedings of the 1st Empirical Methods in Natural Language Processing Conference, 133-141. (1996)

Kerui Chen, Wanli Zuo, Fengling He, Yongheng Chen and Ying Wang

27. Ariel Felner, Sarit Kraus Richard E. Korf.: KBFS: K-best-first search. *Annals of Mathematics and Artificial Intelligence*, Vol. 39, No. 1-2, 19-39. (2003)
28. Giorgos Flouris, Dimitris Plexousakis.: Bridging Ontology Evolution and Belief Change. *Lecture Notes in Computer Science*, Vol. 3955, 486-489. (2006)
29. W. Yang.: Identifying syntactic differences between two programs. *Software Practice and Experience*, Vol. 21, No. 7, 739-755. (1991)
30. WEIFENG SU, JIYING WANG, FREDERICK H. LOCHOVSKY.: ODE: Ontology-Assisted Data Extraction. *ACM Transactions on Database Systems*, Vol. 34, No. 2, 12-35. (2009)

Kerui Chen was born in 1983. She is a Ph.D. in the college of Computer Science and Technology at Jilin University. She current research interests include Web Intelligence, Ontology Engineering and Information integration.

Wanli Zuo was born in 1957. He is a professor and doctoral supervisor at Department of Computer Science and technology, Jilin University. Main research area covers Database Theory, Machine Learning, Data Mining and Web Mining, Web Search Engines, Web Intelligence. He was as a senior visiting scholar, engaged in collaborative research in Louisiana State University (USA) from 1996-1997. He was principle responsible member of 3 national NSFC programs. More than 130 papers of him were published in key Chinese journals or international conferences, 65 of which are cited by SCI/EI. He has five books published by the Higher Education Press of China. And he obtained 5 national and departmental awards. He is a member of System Software Committee of China's Computer Federation, prominent young and middle-aged specialist of Jilin Province.

Fengling He was born in 1962. He is a professor at the Jilin University and a CCF senior member. His research areas are database, data mining and Web search engine.

Yongheng Chen was born in 1979. He is a Ph.D. candidate in the college of Computer Science and Technology at Jilin University. He current research interests include Multi-Core, Database.

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.

Received: October 11, 2010; Accepted: January 19, 2011.