www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 6 June 2015, Page No. 12544-12549

Unit Level Web Databases Searching

Gowri P * *Konda Reddy R*
Dept. of CSE, PBR VITS, Kavali, India
gowri08594@gmail.com
Dept. of CSE, PBR VITS, Kavali, India
rkondareddy75@gmail.com

Abstract— An increasing number of databases have become web accessible through HTML form-based search interfaces. The data units returned from the underlying database are usually encoded into the result pages dynamically for human browsing. We present an automatic annotation approach that first aligns the data units on a result page into different groups such that the data in the same group have the same semantic. Then, for each group we annotate it from different aspects and aggregate the different annotations to predict a final annotation label for it. An annotation wrapper for the search site is automatically constructed and can be used to annotate new result pages from the same web database.

Key words— Data alignment, Data annotation, Web database, Wrapper generation, Annotator

I. INTRODUCTION

Web database is a database application designed to be managed and accessed through the internet.A large portion of the deep web is database based, i.e., for many search engines, data encoded in the returned result pages come from the underlying structured databases. Such type of search engines is often referred as Web databases (WDBs). A typical result page returned from a WDB has multiple search result records (SRRs). Each SRR contains multiple data units each of which describes one aspect of a real-world entity. In this paper, a data unit is a piece of text that semantically represents one concept of an entity. It corresponds to the value of a record under an attribute. It is different from a text node which refers to a sequence of text surrounded by a pair of html tags. Text node is a sequence of text surrounded by pair of html tags. In this paper, we perform data unit level annotation. There is a high demand for collecting data of interest from multiple WDBs. For example, once a book comparison shopping system collects multiple result records from different book sites, it needs to determine whether any two SRRs refer to the same book. The ISBNs can be compared to achieve this. If ISBNs are not available, their titles and authors could be compared. The system also needs to list the prices offered by each site. Thus, the system needs to know the semantic of each data unit. Unfortunately, the semantic labels of data units are often not provided in result pages. Having semantic labels for data units is not only important for the above record linkage task, but also for storing collected SRRs into a-database table (e.g., Deep web crawlers [1]) for later analysis. Early applications require tremendous human efforts to annotate data units manually, which severely limit their scalability. In this paper, we consider how to automatically assign labels to the data units within the SRRs returned from WDBs. Given a set of SRRs that have been extracted from a result page returned from a WDB, our automatic annotation solution consists of three phases. Let d^j_i denote the data unit belonging to the ith SRR of concept j. The SRRs on a result page can be represented in a table format with each row representing an SRR. Phase 1 is the alignment phase. In this phase, we first identify all data units in the SRRs and then organize them into different groups with each group corresponding to a different concept (e.g., all titles are grouped together). The result of this phase with each column containing data units of the same concept across all SRRs. Grouping data units of the same semantic can help identify the common patterns and features among these data units. These common features are the basis of our annotators. In Phase 2 (the annotation phase), we introduce multiple basic annotators with each exploiting one type of features. Every basic annotator is used to produce a label for the units within their group holistically, and a probability model is adopted to determine the most appropriate label for each group. At the end of this phase, a semantic label L^j is assigned to each column. In Phase 3 (the annotation wrapper generation phase), for each identified concept, we generate an annotation rule R^j that describes how to extract the data units of this concept in the result page and what the appropriate semantic label should be. The rules for all aligned groups, collectively, form the annotation wrapper for the corresponding WDB, which can be used to directly annotate the data retrieved from the same WDB in response to new queries without the need to perform the alignment and annotation phases again.

II. LITERATURE SURVEY

In Wrapper Induction System, label the marked data at the same time and then the system can induce rules to extract the same information from same source. It has high extraction accuracy and suffers poor scalability. It also needs to extract from more web sources. In Conceptual Model Based Data Extraction from Multiple-Record WebPages, automatically extract data in multi record and label them. Different domains must be constructed manually .i.e., domain dependent. It is not fully automatics. Wrapper is used data extraction only not for annotation. Automatic Annotation of Data Extracted from Large Websites, it can annotate data units with the closest labels on result pages. But it also has limited applicability, because many web databases don't encode data units with their labels. Ontology assisted Data Extraction System, after labeling the data values with the same label are naturally aligned. It is sensitive to quality and completeness. Vision based approach for deep web Data Extraction, uses visual feature on result pages for alignment, but its alignment is only at text node level. Data Extraction & Label Assignment (DELA) for WDBs uses html tags to align data units by filling them into a table through regular expressions based data tree algorithm.

III. EXISTING SYSTEM

In this existing system, a data unit is a piece of text that semantically represents one concept of an entity. It corresponds to the value of a record under an attribute. It is different from a text node which refers to a sequence of text surrounded by a pair of html tags. It describes the relationships between text nodes and data units in detail. In this paper, we perform data unit level annotation. There is a high demand for collecting data of interest from multiple WDBs. For example, once a book comparison shopping system collects multiple result records from different book sites, it needs to determine whether any two SRRs refer to the same book.

The limitations of existing system are If ISBNs are not available, their titles and authors could be compared. The system also needs to list the prices offered by each site. Thus, the system needs to know the semantic of each data unit. Unfortunately, the semantic labels of data units are often not provided in result pages. For instance, no semantic labels for the values of title, author, publisher, etc., are given. Having semantic labels for data units is not only important for the above record linkage task, but also for storing collected SRRs into a database table.

IV. PROPOSED SYSTEM

In this paper, we consider how to automatically assign labels to the data units within the SRRs returned from WDBs. Given a set of SRRs that have been extracted from a result page returned from a WDB, our automatic annotation solution consists of three phases. This paper has the following contributions:

1) While most existing approaches simply assign labels to each HTML text node, we thoroughly analyze the relationships between text nodes and data units. We perform data unit level annotation.

2) We propose a clustering-based shifting technique to align data units into different groups so that the data units inside the same group have the same semantic. Instead of using only the DOM tree or other HTML tag tree structures of the SRRs to align the data units (like most current methods do), our approach also considers other important features shared among data units, such as their data types (DT), data contents (DC), presentation styles (PS), and adjacency (AD) information.

3) We utilize the integrated interface schema (IIS) over multiple WDBs in the same domain to enhance data unit annotation. To the best of our knowledge, we are the first to utilize IIS for annotating SRRs.

4) We employ six basic annotators; each annotator can independently assign labels to data units based on certain features of the data units. We also employ a probabilistic model to combine the results from different annotators into a single label. This model is highly flexible so that the existing basic annotators may be modified and new annotators may be added easily without affecting the operation of other annotators.

5) We construct an annotation wrapper for any given WDB. The wrapper can be applied to efficiently annotating the SRRs retrieved from the same WDB with new queries.

Our Automatic Annotation Solution consists of three phases they are Alignment Phase, Annotation Phase, Annotation Wrapper Generation Phase .the main improvements is relationship between text nodes and data units are defined and it can explain the alignment algorithm and cluster shifting algorithm.

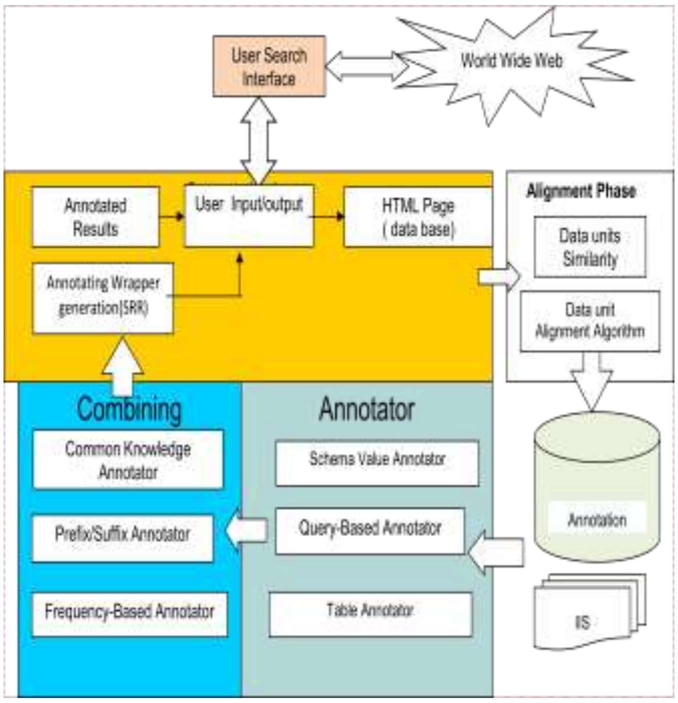

Fig. 1 Block diagram of Unit-level Web databases searching

A. Alignment Phase

In the alignment phase we have data alignment. The purpose of data alignment is to put the data units of the same concept into one group so that they can be annotated holistically. Whether two data units belong to the same concept is determined by how similar they are based on the features described. In this paper, the similarity between two data units (or two text nodes) d_1 and d_2 is a weighted sum of the similarities of the five features between them.

1) Data content similarity (SimC): It is the Cosine similarity [2] between the term frequency vectors of d_1 and d_2.

2) Presentation style similarity (SimP): It is the average of the style feature scores (FS) over all six presentation style features (F) between d_1 and d_2.

3) Data type similarity (SimD): It is determined by the common sequence of the component data types between two data units. The longest common sequence (LCS) cannot be longer than the number of component data types in these two data units. Thus, let t_1 and t_2 be the sequences of the data types of d_1 and d_2, respectively, and TLen (t) represent the number of component types of data type t, the data type similarity between data units' d_1 and d_2.

4) Tag path similarity (SimT): This is the edit distance (EDT) between the tag paths of two data units. The edit distance here refers to the number of insertions and deletions of tags needed to transform one tag path into the other. It can be seen that the maximum number of possible operations needed is the total number of tags in the two tag paths. Let p_1 and p_2 be the tag paths of d_1 and d_2, respectively, and PLen (p) denote the number of tags in tag path p, the tag path similarity between d_1 and d_2.

5) Adjacency similarity (SimA): The adjacency similarity between two data units d_1 and d_2 is the average of the similarity between d^p_1 and d^p_2 and the similarity between d^s_1 and d^s_2.

Our alignment algorithm also needs the similarity between two data unit groups where each group is a collection of data units. We define the similarity between groups G_1 and G_2 to be the average of the similarities between every data unit in G_1 and every data unit in G_2. Our data alignment algorithm is based on the assumption that attributes appear in the same order across all SRRs on the same result page, although the SRRs may contain different sets of attributes (due to missing values). This is true in general because the SRRs from the same WDB are normally generated by the same template program.

Our data alignment method consists of the following four steps. The detail of each step will be provided below.

Step 1: Merge text nodes. This step detects and removes decorative tags from each SRR to allow the text nodes corresponding to the same attribute (separated by decorative tags) to be merged into a single text node.

Step 2: Align text nodes. This step aligns text nodes into groups so that eventually each group contains the text nodes with the same concept (for atomic nodes) or the same set of concepts (for composite nodes).

Step 3: Split (composite) text nodes. This step aims to split the "values" in composite text nodes into individual data units. This step is carried out based on the text nodes in the same group holistically. A group whose "values" need to be split is called a composite group.

Step 4: Align data units. This step is to separate each composite group into multiple aligned groups with each containing the data units of the same concept.

B. Annotation Phase

1) Local versus Integrated Interface Schemas: For a WDB, its search interface often contains some attributes of the underlying data. We denote a LIS as Si = {A_1; A2; . . .; A_k}, where each A_j is an attribute. When a query is submitted against the search interface, the entities in the returned results also have a certain hidden schema, denoted as S_e = {a_1; a_2; . . . ; a_n}, where each a_j (j= 1 . . . n) is an attribute to be discovered. The schema of the retrieved data and the LIS usually share a significant number of attributes. This observation provides the basis for some of our basic annotators. If an attribute at in the search results has a matched attribute A_t in the LIS, all the data units identified with at can be labeled by the name of A_t. However, it is quite often that S_e is not entirely contained in S_i because some attributes of the underlying database are not suitable or needed for specifying query conditions as determined by the developer of the WDB, and these attributes would not be included in Si. This phenomenon raises a problem called local interface schema inadequacy problem. Specifically, it is possible that a hidden attribute discovered in the search result schema Se does not have a matching attribute At in the LIS Si. In this case, there will be no label in the search interface that can be assigned to the discovered data units of this attribute. Another potential problem associated with using LISs for annotation is the inconsistent label problem, i.e., different labels are assigned to semantically identical data units returned from different WDBs because different LISs may give different names to the same attribute. This can cause problem when using the annotated data collected from different WDBs, e.g., for data integration applications. For each WDB in a given domain, our annotation method uses both the LIS of the WDB and the IIS of the domain to annotate the retrieved data from this WDB. Using IISs has two major advantages. First, it has the potential to increase the annotation recall. Since the IIS contains the attributes in all the LISs, it has a better chance that an attribute discovered from the returned results has a matching attribute in the IIS even though it has no matching attribute in the LIS. Second, when an annotator discovers a label for a group of data units, the label will be replaced by its corresponding global attribute name (if any) in the IIS by looking up the attribute-mapping table so that the data units of the same concept across different WDBs will have the same label. We should point out that even though using the IIS can significantly alleviate both the local interface schema inadequacy problem and the inconsistent label problem, it cannot solve them completely. For the first problem, it is still possible that some attributes of the underlying entities do not appear in any local interface, and as a result, such attributes will not appear in the IIS. As to the second problem if one or more of these annotations are not local attribute names in the attribute mapping table for this domain, then using the IIS cannot solve the problem and new techniques are needed.

2) Basic Annotator:

1. Table Annotator (TA): Many WDBs use a table to organize the returned SRRs. In the table, each row represents an SRR. The table header, which indicates the meaning of each column, is usually located at the top of the table. Shows an example of SRRs presented in a table format. Usually, the data units of the same concepts are well aligned with its corresponding column header. This special feature of the table layout can be utilized to annotate the SRRs. Since the physical position information of each data unit is obtained during SRR extraction, we can utilize the information to associate each data unit with its corresponding header. Our Table Annotator works as follows: First, it identifies all the column headers of the table. Second, for each SRR, it takes a data unit in a cell and selects the column header whose area (determined by coordinates) has the maximum vertical overlap (i.e., based on the x-axis) with the cell. This unit is then assigned with this column header and labelled by the header text A (actually by its corresponding global name gn(A) if gn(A) exists). The remaining data units are processed similarly. In case that the table header is not provided or is not successfully extracted by ViNTs [3], the Table Annotator will not be applied.

2. Query-Based Annotator (QA): The basic idea of this annotator is that the returned SRRs from a WDB are always

related to the specified query. Specifically, the query terms entered in the search attributes on the local search interface of the WDB will most likely appear in some retrieved SRRs. A query term "machine" is submitted through the Title field on the search interface of the WDB and all three titles of the returned SRRs contain this query term. Thus, we can use the name of search field Title to annotate the title values of these SRRs. In general, query terms against an attribute may be entered to a textbox or Chosen from a selection list on the local search interface. Our Query-based Annotator works as follows: Given a query with a set of query terms submitted against an attribute A on the local search interface, first find the group that has the largest total occurrences of these query terms and then assign gn(A) as the label to the group.

3. Schema Value Annotator (SA): Many attributes on a search interface have predefined values on the interface. For example, the attribute Publishers may have a set of predefined values (i.e., publishers) in its selection list. More attributes in the IIS tend to have predefined values and these attributes are likely to have more such values than those in LISs, because when attributes from multiple interfaces are integrated, their values are also combined [4]. Our schema value annotator utilizes the combined value set to perform annotation. Given a group of data units $G_{i=}$ {d_1; . . ; d_n} the schema value annotator is to discover the best matched attribute to the group from the IIS. Let A_j be an attribute containing a list of values {v_1; . . . ; v_m } in the IIS. For each data unit d_k, this annotator first computes the Cosine similarities between d_k and all values in A_j to find the value (say v_t) with the highest similarity. Then, the data fusion function CombMNZ [5] is applied to the similarities for all the data units. More specifically, the annotator sums up the similarities and multiplies the sum by the number of nonzero similarities. This final value is treated as the matching score between G_i and A_j. The schema value annotator first identifies the attribute A_j that has the highest matching score among all attributes and then uses gn(A_j) to annotate the group G_i. Note that multiplying the above sum by the number of nonzero similarities is to give preference to attributes that have more matches (i.e., having nonzero similarities) over those that have fewer matches. This is found to be very effective in improving the retrieval effectiveness of combination systems in information retrieval.

4. Frequency-Based Annotator (FA): In Frequency Based Annotator, consider one example "Our Price" appears in the three records and the followed price values are all different in these records. In other words, the adjacent units have different occurrence frequencies. As argued in [1], the data units with the higher frequency are likely to be attribute names, as part of the template program for generating records, while the data units with the lower frequency most probably come from databases as embedded values. Following this argument, "Our Price" can be recognized as the label of the value immediately following it. The phenomenon described in this example is widely observable on result pages returned by many WDBs and our frequency-based annotator is designed to exploit this phenomenon. Consider a group G_i whose data units have a lower frequency. The frequency-based annotator intends to find common preceding units shared by all the data units of the group G_i. This can be easily conducted by following their preceding chains recursively until the encountered data units are different. All found preceding units are concatenated to form the label for the group G_i.

5. In-Text Prefix/Suffix Annotator (IA): In some cases, a piece of data is encoded with its label to form a single unit without any obvious separator between the label and the value, but it contains both the label and the value. Such nodes may occur in all or multiple SRRs. After data alignment, all such nodes would be aligned together to form a group. For example, after alignment, one group may contain three data units, {"You Save $9.50," "You Save $11.04," "You Save $4.45"}. The in-text prefix/suffix annotator checks whether all data units in the aligned group share the same prefix or suffix. If the same prefix is confirmed and it is not a delimiter, then it is removed from all the data units in the group and is used as the label to annotate values following it. If the same suffix is identified and if the number of data units having the same suffix match the number of data units inside the next group, the suffix is used to annotate the data units inside the next group. In the above example, the label "You save" will be assigned to the group of prices. Any group whose data unit texts are completely identical is not considered by this annotator.

6. Common Knowledge Annotator (CA): Some data units on the result page are self-explanatory because of the common knowledge shared by human beings. For example, "in stock" and "out of stock" occur in many SRRs from e-commerce sites. Human users understand that it is about the availability of the product because this is common knowledge. So our common knowledge annotator tries to exploit this situation by using some predefined common concepts. Each common concept contains a label and a set of patterns or values. For example, a country concept has a label "country" and a set of values such as "U.S.A.," "Canada," and so on. As another example, the e-mail address (assume all lower cases) concept has the pattern {a-z 0-9.-%+-]+ @([a-z0 - 9-] +\.)+ [a-z]{2, 4}. Given a group of data units from the alignment step, if all the data units match the pattern or value of a concept, the label of this concept is assigned to the data units of this group. DeLa [6] also uses some conventions to annotate data units. However, it only considers certain patterns. Our Common knowledge annotator considers both patterns and certain value sets such as the set of countries. It should be pointed out that our common concepts are different from the ontologies that are widely used in some works in Semantic Web. First, our common concepts are domain independent. Second, they can be obtained from existing information resources with little additional human effort.

TABLE I
Applicability and Success Rates of Annotators

ANNOTATOR	APPLICABILITY	SUCCESS RATE
Table Annotator	6%	1.0
Query Based Annotator	35%	0.95
Schema Value Annotator	14%	0.5
Frequency Based Annotator	41%	0.86
In-Text Prefix/Suffix Annotator	7%	0.85
Common Knowledge Annotator	25%	0.84

3) Combining Annotator: Our analysis indicates that no single annotator is capable of fully labelling all the data units on different result pages. The applicability of an annotator is the percentage of the attributes to which the annotator can be applied. For example, if out of 10 attributes, four appear in tables, then the applicability of the table annotator is 40 percent. Table shows the average applicability of each basic annotator across all testing domains in our data set. This indicates that the results of different basic annotators should be combined in order to annotate a higher percentage of data units. Moreover, different annotators may produce different labels for a given group of data units. Therefore, we need a method to select the most suitable one for the group. To obtain the success rate of an annotator, we use the annotator to annotate every result page in a training data set. The training result is listed in Table. It can be seen that the table annotator is 100 percent correct when applicable. The query-based annotator also has very high success rate while the schema value annotator is the least accurate. An important issue DeLa did not address is what if multiple heuristics can be applied to a data unit. In our solution, if multiple labels are predicted for a group of data units by different annotators, we compute the combined probability for each label based on the annotators that identified the label, and select the label with the largest combined probability. One advantage of this model is its high flexibility in the sense that when an existing annotator is modified or a new annotator is added in, all we need is to obtain the applicability and success rate of this new/revised annotator while keeping all remaining annotators unchanged. We also note that no domain-specific training is needed to obtain the applicability and success rate of each annotator.

C. Annotation Wrapper Generation Phase

Each annotator group of data units corresponds to an attribute in the SRRs. Annotation Wrapper is a description of the annotation rules for all attributes on the result page. To use the wrapper to annotate a new result page for each data unit in an SRR, the annotation rules are applied on it. Once the data units on a result page have been annotated, we use these annotated data units to construct an annotation wrapper for the WDB so that the new SRRs retrieved from the same WDB can be annotated using this wrapper quickly without reapplying the entire annotation process. We now describe our method for constructing such a wrapper below. Each annotated group of data units corresponds to an attribute in the SRRs. The annotation wrapper is a description of the annotation rules for all the attributes on the result page. After the data unit groups are annotated, they are organized based on the order of its data units in the original SRRs. To use the wrapper to annotate a new result page, for each data unit in an SRR, the annotation rules are applied on it one by one based on the order they appear in the wrapper. If this data unit has the same prefix and suffix as specified in the rule, the rule is matched and the unit is labelled with the given label in the rule. If the separators are specified, they are used to split the unit, and $label_i$ is assigned to the unit at the position $unitindex_i$.

IV. RESULTS

The optimal feature weights obtained through our genetic training method over DS1 are {0.64, 0.81, 1.0, 0.48, and 0.56} for SimC, SimP, SimD, SimT, and SimA, respectively, and 0.59 for clustering threshold T. The average alignment precision and recall are converged at about 97 percent. The learning result shows that the data type and the presentation style are the most important features in our alignment method. Then, we apply our annotation method on DS1 to determine the success rate of each annotator. Table shows the performance of our data alignment algorithm for all 90 pages in DS2. The precision and recall for every domain are above 95 percent, and the average precision and recall across all domains are above 98 percent. The performance is consistent with that obtained over the training set. The errors usually happen in the following cases. First, some composite text nodes failed to be split into correct data units when no explicit separators can be identified. For example, the data units in some composite text nodes are separated by blank spaces created by consecutive HTML entities like “&nbsp;” or some formatting HTML tags such as <SPAN>. Second, the data units of the same attribute across different SRRs may sometimes vary a lot in terms of appearance or layout. For example, the promotion price information often has color or font type different from that for the regular price information. Note that in this case, such two price data units have low similarity on content, presentation style, and the tag path.

Even though they share the same data type, the overall similarity between them would still be low. Finally, the decorative tag detection (Step 1 of the alignment algorithm) is not perfect (accuracy about 90 percent), which results in some tags to be falsely detected as decorative tags, leading to incorrect merging of the values of different attributes. We will address these issues in the future. We also conducted experiments to evaluate the significance of each feature on the performance of our alignment algorithm. For this purpose, we compare the performance when a feature is used with that when it is not used. Each time one feature is selected not to be used, and its weight is proportionally distributed to other features based on the ratios of their weights to the total weight of the used features.

The alignment process then uses the new parameters to run on DS2. It can be seen that when any one of these features is not used, both the precision and recall decrease, indicating that all the features in our approach are valid and useful. We can also see that the data type and the presentation style are the most important features because when they are not used, the precision and recall drop the most (around 28 and 23 percentage points for precision, and 31 and 25 percentage points for recall, respectively). This result is consistent with our training result where the data type and the presentation style have the highest feature weights.

V. CONCLUSION

In this study, the data annotation problem and proposed a multi-annotator approach to automatically constructing an annotation wrapper for annotating the search result records retrieved from any given web database. This approach consists of six basic annotators and a probabilistic method to combine the basic annotators. Each of these annotators exploits one type of features for annotation and our experimental results show that each of the annotators is useful and they together are capable of generating high quality annotation. A special feature of our method is that, when annotating the results retrieved from a web database, it utilizes both the LIS of the web database and the IIS of multiple web databases in the same domain. We also explained how the use of the IIS can help alleviate the local interface schema inadequacy problem and the inconsistent label problem. Accurate alignment is critical to achieving holistic and accurate annotation. Our method is a clustering based shifting method utilizing richer yet automatically obtainable features. This method is capable of handling a variety of relationships between HTML text nodes and data units, including one-to-one, one-to-many, many-to-one, and one-to-nothing. Our experimental results show that the precision and recall of this method are both above 98 percent. There is still room for improvement in several areas. For example, we need to enhance our method to split composite text node when there are no explicit separators. We would also like to try using different machine learning techniques and using more sample pages from each training site to obtain the feature weights so that we can identify the best technique to the data alignment problem.

REFERENCES

[1] Yiyao Lu, Hai He, Hongkun Zhao, Weiyi Meng, and Clement Yu, *Annotating Search Results from Web Databases*,
IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 3, Mar. 2013.

[2] N. Krushmerick, D. Weld, and R. Doorenbos, *Wrapper Induction for Information Extraction*, Proc. Int’l Joint Conf.
Artificial Intelligence (IJCAI), 1997.

[3] Embley, D. Campbell, Y. Jiang, S. Liddle, D. Lonsdale, Y. Ng, and R. Smith, *Conceptual-Model-Based Data*
Extraction from Multiple-Record Web Pages, Data and Knowledge Eng., vol. 31, no. 3, pp. 227-251, 1999.

[4] V. Crescenzi, G. Mecca, and P. Merialdo, *RoadRUNNER: Towards Automatic Data Extraction from Large Web*
Sites, Proc. Very Large Data Bases (VLDB) Conf., 2001.

[5] W. Liu, X. Meng, and W. Meng, *ViDE: A Vision-Based Approach for Deep Web Data Extraction*, IEEE Trans.
Knowledge and Data Eng., vol. 22, no. 3, pp. 447-460, Mar. 2010.

[6] J. Wang and F.H. Lochovsky, *Data Extraction and Label Assignment for Web Databases*, Proc. 12th Int’l Conf.
World Wide Web (WWW), 2003.