Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Efficient Mechanism for Dynamic Metadata Search Record for Web Database

Prasad B. Dhore, Rajesh B. Singh
Department of Computer Engineering
Sinhgad Institute of Technology
Lonavala, Pune, Maharashtra, India

Abstract: This paper investigates techniques for extracting data from HTML sites through the use of automatically generated wrappers. 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. For the encoded data units to be machine process able, which is essential for many applications such as deep web data collection and Internet comparison shopping, they need to be extracted out and assigned meaningful labels To automate the wrapper generation and the data extraction process, the paper develops and 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. So our experiments indicate that the proposed approach is highly effective.

Keywords - Annotation, HTML, wrapper.

I. INTRODUCTION

Data extraction from HTML is usually performed by software modules called wrappers. A key problem with manually coded wrappers is that writing them is usually a difficult and labor intensive task and that by their nature wrappers tend to be brittle and difficult to maintain. 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 (WDB).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. There is a high demand for collecting data of interest from multiple WDBs. 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. 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.

While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks.

II. RELATED WORK

L. Arlotta, V. Crescenzi, G. Mecca, and P. Meri- aldo proposed Automatic Annotation of Data Extracted from Large Web Sites [1]. this paper can automatic web record extraction set of object from heterogeneous web page bases an object in automated fashion

V. Crescenzi, G. Mecca, and P. Merialdo proposed Roadrunner Towards Au-tomatic Data Extraction from Large Web Sites [2]. In this paper, investigate technique for extract- ing data from HTML sites through the use of automatically generate wrappers

Saradha Arvindhan.r proposed on map reducing for annotation of search result from web database [3], HTML from web database interface make large number of database web accessible. when user submit query to search engine, data unit are retrived from underlying database.

V..Kalyan Deepak and n.v.rajeesh kumar proposed the retrive record from web database using data alignment [4]. In this research paper data unit are return from underlying database are usually encoded into result page dynamically for human browsing.

Luigi Arlotta and Valter Crescenzi proposed Automatic Annotation of Data Extracted from Large Web Sites [5] These systems are based on unsupervised inference techniques: taking as input a small set of sample pages, they can produce a common wrapper to extract relevant data. However, due to the automatic nature of the approach, the data extracted by these wrappers have anonymous names. In the framework of our ongoing project RoadRunner, we have developed a prototype, called Labeller that automatically annotates data extracted by automatically generated wrappers.

III. INPUT DESIGN AND OUTPUT DESIGN

A. INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple.

OBJECTIVES:

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user.

B. OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

OBJECTIVES:

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information
3. Create document, report, or other formats that contain information produced by the system.

IV. PROPOSED METHODOLOGY AND ARCHITECTURE

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 and 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.

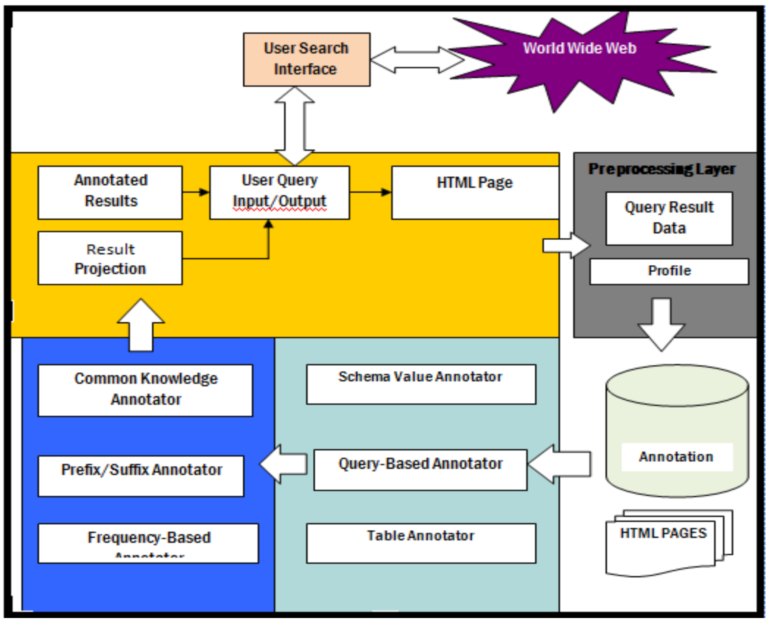

Fig 1. Proposed System Architecture

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 and 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.

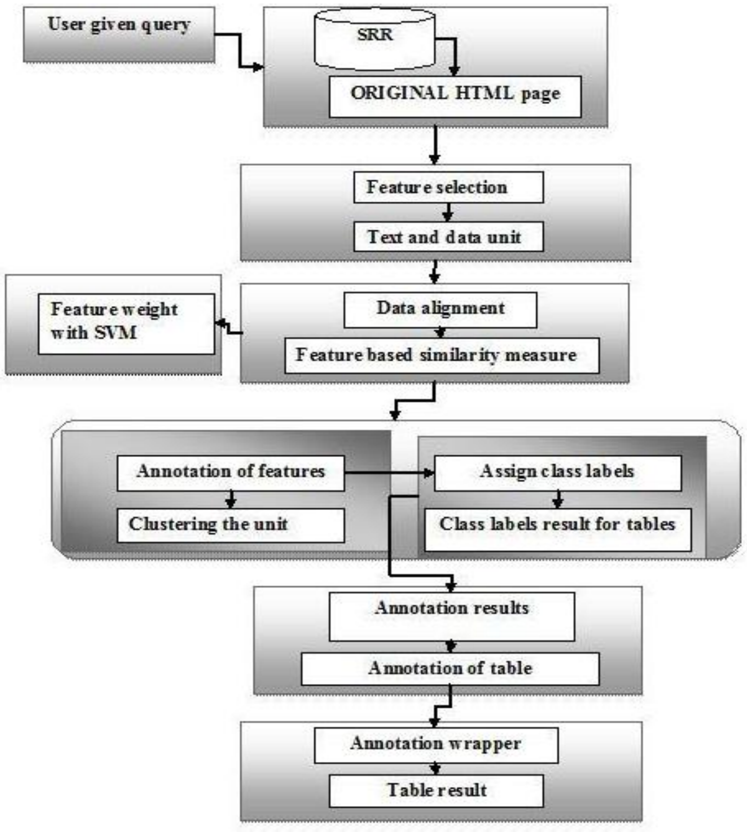

Fig 2. Process of System Architecture

V. IMPLEMENTATION MODULES

A. *Basic Annotators*

In a returned result page containing multiple SRRs, the data units corresponding to the same concept (attribute) often share special common features. And such common features are usually associated with the data units on the result page in certain patterns. Based on this observation, we define six basic annotators to label data units, with each of them considering a special type of patterns/features. Four of these annotators (i.e., table annotator, query-based annotator, in text prefix/suffix annotator, and common knowledge annotator) are similar to the annotation heuristics

B. *Query-Based Annotator*

The basic idea of this annotator is that the returned SRRs from a WDBare 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. For example, 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.

C. *Schema Value Annotator*

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. Our schema value annotator utilizes the combined value set to perform annotation. The schema value annotator first identifies the attribute Aj that has the highest matching score among all attributes and then uses gn(Aj) to annotate the group Gi. 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

D. Common Knowledge Annotator

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. 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.

E. Combining Annotators

Our analysis indicates that no single annotator is capable of fully labeling 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. 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. Our annotators are fairly independent from each other since each exploits an independent feature.

VI. MATHEMATICAL MODULE AND ALGORITHM

A. MATHEMATICAL MODULE

1..*Data content similarity (SimC).* It is the Cosine similarity between the term frequency vectors of d1 and d2:

$SimC(d1,d2) = (vd1*vd2)/(||vd1||*||vd2||)$ (1)

where Vd is the frequency vector of the terms inside data unit d, ||Vd|| is the length of V_d, and the numerator is the inner product of two vectors.

2.*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 t1 and t2 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 d1 and d2 is

$SimD(d1,d2) = LCS(t1,t2)/ MAX\ Tlen(t1),Tlen(t2)$ (2)

3.*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 p1 and p2 be the tag paths of d1 and d2, respectively, and PLen(p) denote the number of tags in tag path p, the tag path similarity between d1 and d2 is

$SimT(d1,d2) = 1- EDT(p1,p2)/ PLen(p1)+PLen(p2)$ (3)

4. *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

$SimP(d_1,d_2) = \sum_{i-1}^{6} FS_i / 6$ (4)

B. ALIGNMENT ALGORITHM

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 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.

Step 4: Composite text nodes. aligns text nodes into groups so that eventually each group contains the text nodes with the same concept. A group whose “values” need to be split is called a composite group

Step 5: 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.

```
ALIGN (RRs):
1.      b←1;
2.      While true
3.      for a←1 to number of RRs
4.      GRP b←RR[a][b];
5.      if GRP b is empty
6.      exit;
```

```
7.  V←CLUSTERING(GRP);
8.  if |V| >1
9.  S←Ø;
10. for P←1 to number of RRs
11. for Q←b+1 to RR[a] length
12. S< RR[p][q];
13. V[c]=min(sim(V[c],S));
14. for c←1 to |v| and c≠c
15. foreach RR[p][b] in V[C]
16. insert NIL at position b in RR[p];
17. b←b+1;
```

Algorithm for aligning the data units
CLUSTERING (GRP):

```
1.  V←all data units in GRP
2.  while |V| >1
3.  best←0;
4.  L←NIL; R←NIL;
5.  foreach X in V
6.  foreach Y in V
7.  if((X!=Y) and (sim(X,Y)>best))
8.  best←sim(X,Y);
9.  L←X;
10. R←Y;
11. if best > T
12. remove L from V;
13. remove R from V;
14. add L U R to V;
15. else break loop;
16. return V;
```

VII. CONCLUSIONS

In this paper, we studied 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 and studied the automatic data alignment 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] A. Arasu and H. Garcia-Molina "Extracting Structured Data from Web Pages," Proc. SIGMOD Int'l Conf. Management of Data, 2003.
[2] L. Arlotta, V. Crescenzi, G. Mecca, and P. Merialdo, "Automatic Annotation of Data Extracted from Large Web Sites," Proc. Sixth Int'l Workshop the Web and Databases (Web DB), 2003.
[3] P. Chan and S. Stolfo, "Experiments on Multistrategy Learning by Meta-Learning," Proc. Second Int'l Conf. Information and Knowledge Management (CIKM), 1993.
[4] W. Bruce Croft, "Combining Approaches for Information Retrieval," Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval, Kluwer Academic, 2000.
[5] V. Crescenzi, G. Mecca, and P. Meridio, "RoadRUNNER: Towards Automatic Data Extraction from Large Web Sites," Proc. Very Large Data Bases (VLDB) Conf., 2001.
[6] S. Dill et al., "SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation," Proc. 12th Int'l Conf. World Wide Web (WWW) Conf., 2003.

[7] H. Elmeleegy, J. Madhavan, and A. Halevy, “Harvesting Relational Tables from Lists on the Web,” Proc. Very Large Databases (VLDB) Conf., 2009.

[8] D. 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.

[9] D. Freitag, “Multistrategy Learning for Information Extraction,” Proc. 15th Int’l Conf. Machine Learning (ICML), 1998.

[10] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, 1989.