ONTOLOGY BASED DATA EXTRACTION AND ALIGNMENT USING NESTED MODEL

L. Bharathi Krishna #1 G. Sangeetha *2

#1 Assistant Professor, Department of Computer Science and Engineering
Apollo Engineering College, Chennai, Tamil Nadu, India

*2 PG Scholar, Department of Computer Science and Engineering
Apollo Engineering College, Chennai, Tamil Nadu, India

ABSTRACT

Automatic data extraction is very important for many applications that need to co-operate with multiple web databases. Ontology assisted data extraction is proposed, which have two phases namely ontology construction and data extraction. Ontology construction modules have four steps. They are primary wrapping, query result annotation, attribute matching and ontology construction step. Primary wrapping extracts the query result records from the query result pages and align them into query result table. Then from the obtained query result table, the query annotator assigns a label for each column of the query result table and query interfaces. Next, each column of the table and query interface is matched by using attribute matcher. Finally the, the ontology is constructed using ontology construction step. Data extraction phase consist of three steps process. First step identifies query result section by constructing the tag tree from the query result pages and also finds the sub tree whose raw data strings have large association with the ontology. Then the query result section is segmented into query result records using record segmentation. For the generated query result records, the data value alignment and label alignment assigns the attribute name for each data value of the QRR.

1. INTRODUCTION

The Internet contains a huge number of information sources of different kinds. Even if a user has the possibility of browsing the Internet, the search of relevant information is still a difficult task. Most search engines use keywords to identify possible answers to a user's query, and return a list of links to the documents. Many of the returned documents are not relevant to the query, and the user often has to browse the returned document list to find relevant information. Although a search engine provides useful help for users to identify relevant information, it cannot be used by a software agent to obtain reliable data to fulfil its tasks. This is mainly due to the lack of precision and standard formalism of the returned answers. In addition, current search engines are more focused on static data on the web.
rather than dynamic data that constantly change, such as weather forecast, stock exchange information, etc. Such data are more and more required by software agents, and the need is growing to find ways to extract data so they can be fully exploited by agents.

Databases accessible on the Web, called *web databases*, compose what is referred to as the *deep web*. Unlike pages in the surface web, which are stored for subsequent querying after they are generated, deep web pages are usually not stored, but are generated dynamically from web databases in response to a user query submitted through a query interface. A web database responds to a user query with the relevant data, either structured or semi-structured, embedded in HTML pages (called *query result pages* in this paper). To utilize this data, it is necessary to extract it from the query result pages. Automatic data extraction is very important for many applications, such as meta-querying, data integration and data warehousing, that need to co-operate with multiple web databases. The goal of data extraction is to remove the irrelevant information from a query result page, extract the query result records (referred to in this paper as *QRRs*) from the page and align the data values in the extracted records into a table so that the data values for the same attribute in each record are put into the same column in the table.

We present a novel data extraction method, ODE (Ontology-assisted Data Extraction), that automatically extracts the QRRs embedded in an HTML page generated by a deep web site. For a given query, ODE uses both the query interfaces and the query result pages of deep web sites from the same domain to automatically construct a domain ontology. In the data value alignment and label assignment steps, the ontology is used to associate each data value in a QRR with an ontology attribute, with the assistance of a maximum entropy model that assigns attribute names to data values using context and tag structure information as features. To our knowledge, we are the first to use a maximum entropy model for data extraction. ODE is fully automatic and overcomes many of the shortcomings of current automatic data extraction methods. Furthermore, experimental results show that ODE is extremely accurate in identifying the query result section in an HTML page, segmenting the query result section into records, and aligning and labeling the data values in the records.

1.1 AIM

The main aim of the proposed system is to remove the irrelevant information from query result page.

1.2 OBJECTIVE

The objectives of the proposed systems are

1. Obtain relevant data for the user query

2. Automatically extracting data records that are encoded in the query result pages generated by web databases.

1.3 GOALS

The main goal of the proposed system is to provide relevant data and extract the data from the query result record
1.4 SCOPE

The scope of the proposed system is to automatically extracting the data from the query result pages of the heterogeneous web site using ontology

2. RELATED WORKS

In recent years, the volume and quality of deep web information has attracted such research attention. As the returned data for a query are embedded in HTML pages, much research has focused on extracting the data from these query result pages. Simultaneously, many researchers have studied the problem of extracting information from HTML files. Earlier works focused on wrapper induction, during which human assistance is required to build the wrapper. Recently, several data extraction methods have been proposed to automatically extract the records from a query result page.

In early work on wrapper induction, extraction rules are semi-automatically derived based on inductive learning. A user labels or marks the data to extract (the target data) in a set of training pages or a list of data records in a page and the system then learns the extraction rules from the labeled data and uses them to extract records from new pages. A rule usually contains two patterns, a prefix pattern and a suffix pattern, to denote the beginning and the end, respectively, of the target data. Some existing wrapper induction systems include WL2 [Cohen et al. 2002], SoftMealy [Hsu and Dung 1998], WIEN [Kushmerick 2000] and Stalker [Muslea at al. 1999]. Semi-automatic wrapper induction has the advantage that no extraneous data are extracted as the user can label only the data in which he/she is interested. Furthermore, these methods are usually very fast at extracting data from web pages, faster than most other kinds of techniques, including ODE. Hence, many real-time applications, such as meta-search, use wrapper induction techniques to extract data from web pages. However, semi-automatic wrapper induction requires labor intensive and time consuming manual labeling of data. Hence, it is not scalable to a large number of web sites. Moreover, an existing wrapper usually performs poorly when the format of a query result page changes. Considering that the Web changes rapidly, the format of a query result page may change frequently. Hence, wrapper induction involves two further difficult problems: page monitoring to determine whether a page’s format has changed and wrapper maintenance to maintain a wrapper when a page’s format changes. To overcome the problems of semi-automatic wrapper induction, some unsupervised learning methods, such as Road Runner [Crescenzi et al. 2001], Omini [Buttler et al. 2001], ExAlg [Arasu and Garcia-Molina 2003], IEPAD [Chang and Lui 2001], DeLa [Wang and Lochovsky 2003] and PickUp [Chen et al. 2004], have been proposed to fully automatically extract the data from a query result page based on the tag structure that exists in one or several HTML pages from the same web site. RoadRunner [Crescenzi et al. 2001] starts with any page as its initial page template and then compares this template with each new page. If the template cannot generate the new page, it is fine-tuned. However, RoadRunner suffers from several limitations: 1. When RoadRunner finds that the current template cannot generate a new page, it searches through an exponential size page schema trying to fine-tune the template. 2. RoadRunner assumes that the template generates all HTML tags, which does not hold for many web databases. 3. RoadRunner assumes that there are no disjunctive attributes, which the authors of RoadRunner admit does not hold for many query result pages. 4. Data labeling/annotation is not addressed in RoadRunner, since it mainly focuses on the data extraction.
problem. Omini [Buttler et al. 2001] uses several heuristics to extract a subtree that contains data strings of interest from an HTML page. Then, another set of heuristics is used to find a separator to segment the minimum data object-rich subtree into data records. However, the results reported in [Liu et al. 2003] indicate that Omini has low effectiveness (recall 39% and precision 56%) for extracting data records. ExAlg [Arasu and Garcia-Molina 2003] works on a set of web pages from the same web site and computes equivalence classes, which are sets of tokens having the same frequency of occurrence on all input pages. Large and frequently occurring equivalence classes are extracted for page template generation. ExAlg has several problems. First, it assumes that a tag or string is used to separate data strings, which is not valid for many web sites. Second, while, ExAlg can handle optional and disjunctive attributes, it cannot handle pages that contain lists. Finally, it is possible that multiple equivalence classes are extracted, which would then require human involvement to select one. IEPAD [Chang and Lui 2001] first encodes all HTML tokens of a parsed web page into a binary sequence and then uses a PAT tree and heuristics to find frequent patterns. The users can choose one of the generalized patterns as an extraction rule. While IEPAD is efficient for web pages that only contain plain-structured data, the approach is not fully automatic and cannot handle nested-structured data with multi-value attributes. DeLa [Wang and Lochovsky 2003] models the data strings contained in template generated web pages as string instances, encoded in HTML tags, of the implied nested type of their web database. A regular expression is employed to model the HTML encoded version of the nested type. Since the HTML tag-structure enclosing a data string may appear repeatedly if the page contains more than one instance of the data string, the page is first transformed into a token sequence composed of HTML tags and a special token “text” representing any text string enclosed by pairs of HTML tags. Then, Crepeated substrings are extracted from the token sequence and a regular expression wrapper is induced from the repeated substrings according to some hierarchical relationships among them. The main problem with this method is that it often produces multiple patterns (rules) and it is hard to decide which is correct. Pick Up [Chen et al. 2004] identifies table structures in web pages by also mining repeated patterns in HTML tag sequences.

TISP [Tao and Embley 2007] constructs wrappers by looking for commonalities and variations in sibling tables in sibling pages (i.e., pages that are from the same web site and have similar structures). Commonalities in sibling tables represent labels, while variations represent data values. Matching sibling tables are found using a tree-mapping algorithm applied to the DOM tree representation of tagged tables in sibling pages. Using several predefined table structure templates, which are described using regular expressions, TISP “fits” the tables into one of the templates allowing the table to be interpreted. TISP is able to handle nested tables as well as variations in table structure (i.e., optional columns) and is able to adjust the predefined templates to account for various combinations of table templates. The whole process is fully automatic and experiments show that TISP achieves an overall F-measure of 94.5% on the experimental dataset. First, since HTML tags are often used in unexpected and unconventional ways, one cannot rely on “proper” HTML tag usage. Second, since the main purpose of HTML tags is to facilitate the rendering of web pages, they convey little semantic information about the data strings. Consequently, an ill-structured HTML page may still display correctly. Furthermore, some data strings may contain embedded tags, which may confuse the wrapper generators making them even less reliable. Recently, to overcome these shortcomings, visual features have been used for data extraction. In DEPTA [Zhai and Liu 2006], the intuition that the gap within a QRR is typically smaller than that between QRRs is used to segment data records and to identify individual data records. The data strings in the records are then aligned,
for those data strings that can be aligned with certainty, based on tree matching. However, ViNTs requires a no-query-result page and multiple QRRs (at least four) in each non-empty query result page to generate an accurate wrapper and none of them do label assignment.

3. PROPOSED SCHEME

Proposed a novel data extraction and alignment method called ODEAN (ontology based data extraction and alignment using nested model). This method automatically extracts and aligns the query result records from the HTML pages. The ontology for a domain is first constructed from query result pages and query interfaces of the web sites, and then used to extract data records from a query result page in the domain. The information matching (within one site or among multiple sites) between a query result page and a query interface also can be very useful to understand the query result pages and has been used in label assignment in multiple data extraction models. The data extraction component extracts the data from the query result pages using the constructed ontology method. After extracting the data we perform data alignment using nested model. In this nested model we combine nested structure and ontology models. After aligning the data we are using labeling model for assigning attribute for each record in the alignment table.

4.1 ADVANTAGES

1. Automatic data extraction and alignment method.
2. Not depends on tag structure.
3. Handle Nested Structure problem
4. Does not discard data regions.
5. Assign Argument label for all records

ARCHITECTURE OF PROPOSED SYSTEM:

system architecture is the conceptual model that defines the structure and behavior of the system. The system architecture for the proposed system ontology based data extraction architecture is given in Figure 3.1
ALGORITHMS

ATTRIBUTE_VALUE_PAIR_GENERATION_ALGORITHM

1: deletes the bad HTML tags and syntactical errors in $P$ and turns the body of $P$ into a DOM tree, $T$.

2: discard HTML attributes and representation tags, such as b, i and font, from $T$

3: for each leaf node $i$ in $T$ do

4: if the content of $i$ matches any keyword in $Kr$ then

5: annotate $i$ with the semantic role $r$

6: if the content of $i$ does not match any keyword then

7: annotate $i$ with the unidentified role

8: if $i$ is annotated with $d$ ($d > 1$) semantic roles then

9: separate $i$ into $d$ nodes, and annotate $d$ nodes with their corresponding semantic roles

10: traverse $T$ in a breadth-first way, and sort all non-leaf nodes of $T$ in the reverse order of the traversal sequence

11: for each non-leaf node $j$ in $T$ do

12: calculate the structural-semantic entropy $ej$ for $j$
13: if $ej \geq Hd$ and $j$ has a greater structural-semantic entropy than all its descendant nodes then 14: $j$ is a data-rich node, and makes all its descendant nodes non data-rich node.

15: if $Hl \leq ej < Hd$ and $j$ is the common parent node of the sibling nodes that have the same nonzero values of structural-semantic entropy then

16: $j$ is a list node

17: if any structural-semantic entropy of the sibling nodes is less than $Hd$ then

18: $j$ is a link offer node

19: for each data-rich node $m$ do

20: for each leaf node $n$ of $m$ that is annotated with a semantic role do

21: extract a value for the semantic role (if the regular expression for matching value is defined, the value should be tested) and associate the value with the corresponding attribute

22: insert the attribute-value pairs of a record into $V$

23: return $V$ end {DE-SSE}

Global Labeling for web database

Step1: consider the number of attributes $Bi= \{b1, b2, b3….bn\}$, where every $b_j$ is an attribute.

Step2: Consider $Be= \{a1, a2, .an\}$ attribute to be discovered from the search interfaces

Step 3: if $(a_j==b_j)$

{  
Label: $b_j=a_j$ ;( create the local attributes from search interface with highest probability)
}

Step 4: THEN Combines all the matched attributes and assign as global matching attribute. Each global attribute has a unique global name

For example $b1=a1+a2+a3$

Else {

No label: Assign unmatched attributes

}

END If

Repeat: step 3 and step 4 until discovered attributes is match with any search interfaces
Choose the highest probability matching attributes

Step 5: If no matching with the search interfaces eliminates the unmatched attributes

Step 6: return global mapping attribute table

**Subtree Identification Algorithm**

\[ \text{SearchDataRegion}(R,O) \]

\{  
T \leftarrow R_i \in R;  
\text{While} \ (T \text{ is not leaf node})  
begin \  
T_i \leftarrow \text{one child node of } T \text{ has the largest correlation with } O; \  
\text{if} (\text{Corr}(T,O) \lt \text{Corr}(T_i,O)) \ T \leftarrow T_i \  
\text{else break}  
End  
\text{While} \ (1)  
Begin \  
T_l \leftarrow \text{the left sibling of } T \  
\text{If} \ (\text{Corr} (T, O) \lt \text{Corr} (T_l+T, O)) \ T \leftarrow T_l+T \  
\text{Else break}  
End  
\text{While} \ (1)  
Begin \  
T_r \leftarrow \text{the right sibling of } T \  
\text{If} (\text{Corr}(T,O) \lt \text{Corr}(T_r+T,O)) \ T \leftarrow T_r+T \  
\text{else break}  
End  
\text{Return } T  
\}
IMPLEMENTATION DETAILS

The proposed system consists of the following modules,

Ontology Construction

1. Tag tree Construction
2. Data Value Assignment for the extracted QRR
3. Label Value Assignment
4. Attribute Matching
5. Ontology Construction
6. Region Identification

3.1 Tag tree Construction:

The main task of the data region construction is to find some instances from a query result pages. For the given query result pages first has to construct HTML tag tree, then from constructed the HTML tag tree, Primary wrapper assumes that similar QRRs are represented as child sub-trees of the same parent node in a tag tree. Then the obtained similar QRRs (Query Result Records) form a data region. Primary wrapper requires that the data region include several sets of similar nodes in the tag tree in order to find the data region. Once the QRR is identified, the next task is to assign the labels to the columns of the data table containing the extracted QRRs. The four heuristics proposed are used to assign the label QRR. The first heuristics is when the user submits keyword query, if the keyword mostly appears in one specific column of the result data table, and then we assign the label of that query interface element to the column using match query interface element labels to the data values.

![Fig 3.1: Tag tree Construction](image)

3.2 Data Value Alignment

Data value alignment and label assignment can be performed simultaneously as a single step. That is, for each data value, ODEN first assign a label to it. Next, the data values with the same label are aligned to the same column in the final table and the shared label assigned as the column’s label.
3.3 Label Value Assignment for the Extracted QRR

Once the QRR is identified, the next task is to assign the labels to the columns of the data table containing the extracted QRRs. The four heuristics proposed are used to assign the label QRR. The first heuristics is when the user submits keyword query, if the keyword mostly appears in one specific column of the result data table, and then we assign the label of that query interface element to the column using match query interface element labels to the data values.

3.4 Attribute Matching

Attribute matching is the next task in the ontology construction. Where attribute matcher matches the columns of the query result table with the user query interface from the various web sites. Three kinds of matching are available they are Value-level matching, Label-level matching and Label-value matching. In Value-level matching has to identify the query result table columns from different web sites using the data value similarity among different web sites. In Label-level matching has to identify the similarity matching among the query result table columns from different web sites according to the label co-occurrence pattern in the query result tables.
3.5. Ontology Construction

From the matching the ontology is build for the particular domain. The nine steps process are used to create the ontology

First step: if the matching is common, then the attribute is created

Second step: if the matching is 1:1, then the attribute is created in which any label matching is randomly selected as the attribute name and other labels are set as aliases.

Third step: If the matching is 1: n, n+1 attributes are created

Fourth step: If the matching is \( n:m \ \{a_1, ..., an\} = \{an+1, ..., an+m\} \), we can treat it as a 1:n matching and a 1:m matching by generating an artificial attribute \( aa \) and two matching to this attribute: \( aa = \{a_1, ..., an\} \) and \( aa = \{an+1, ..., an+m\} \) and process the 1:n matching and 1:m matching using the method described in the 3rd point above. Hence \( n+m+1 \) attributes are created.

Fifth step: The values of an attribute are obtained from the data values that appear in the query result pages from the training web sites.

Sixth step: The data type and external representation of an attribute are derived through the parsing of its values. The parsing starts from some conventional data format, such as date time, int, real, etc., and then try to find the maximal-prefix and maximal-suffix shared by data values with the same label from the same domain.

Seventh step: The value probability of an attribute is calculated as the web-site-average of the average occurrence probability of a data value in a QRR within each training web site.

Eighth step: The name probability of an attribute is measured as the web-site-average of the average occurrence probability of the attribute name in a QRR in each training web site.

Ninth step: has to find the max occurrence count of the attribute value in all QRRs of the training web sites.
3.6. Region Identification

After completion of ontology construction, has to identify the query result section using the primary wrapper. It forms the query result section when the two or more QRR appear in the query result page. Also the QRRs are in the same format and are separated in different regions will be combined into the same query result section. On the other hand have to find whether the instances in the query result section returned by primary wrapper are the actual QRRs or not.

1. First step is identifying whether there is another query result section that contains QRRs and return the query result section if it exists.

2. Second step is assumes that there is only one QRR in the query result page and finds the subtree with the largest correlation with the ontology, using the algorithm maximum correlation subtree identification.

3. Third step is to consider that there is no QRR in the query result page.

CONCLUSION

The problems of data extraction from web pages deal in the proposed system. In particular, our goal is to find a way to extract reliable data, and to convert them in a standard form. The approach described fully automates wrapper generation for Web documents that are rich in data, narrow in ontological breadth, and have multiple records on a single page. Instead of using page structure or
HTML format as a guide to extracting data. We used a predefined ontological model instance for the chosen application. An application ontology provided the relationships among the objects of interest, the cardinality constraints for these relationships, a description of the possible strings that can populate various sets of objects, and possible context keywords to help match values with object sets. To prepare unstructured documents for comparison with the ontology, we provided a means to identify the records of interest on a Web page. With the ontology and record extractor in place, we automatically extracted records and fed them to a processor that heuristically matched them with the ontology to populate a database with the extracted data.

REFERENCES


