Combine Tag and Value Similarity for Data Extraction and Alignment

D.Phani Sri Lakshmi
Student
Dept. of CSE
SVECW
Bhimavaram, India

D.N.S.B.Kavitha
Assistant professor
Dept. of CSE
SVECW
Bhimavaram, India

Abstract

Based on a user's query Web databases create query result pages. For many applications, such as data integration, which needs to cooperate with multiple web databases there, is a need to automatically extract the data from these query result pages .So we present a data extraction and alignment method called CTVS which combines both tag and value similarity. The data values from the same attribute are put into the similar column in which CTVS automatically extract data from query result pages by first identifying and segmenting the query result records (QRRs) in query result pages and then align the segmented QRRs into a table. Specially, we advise new techniques to switch the case when the QRRs are not secure, which may be due to the presence of main information, such as a commentary, proposal or advert, and for handling any nested structure that may exist in the QRRs. By CTVS, we create novel record alignment algorithms that align the attributes in a record, in pair wise first and then holistically. Experimental l results show that CTVS achieves high precision and outperforms alive state-of-the-art data extraction methods.

Keywords — Automatic wrapper generation, data extraction, data record alignment, information integration, nested structure processing.

1. INTRODUCTION

ONLINE database, called web database, include pages in the deep web are dynamically generate in response to a user query submitted through the query interface of a web database, which Compared with webpages in the surface web, that can be accessed by a unique URL. Upon receiving a user's query, a web database returns the related data, either structured or semi structured, encoded in HTML pages. Many web applications need the data from multiple web databases, such as meta querying, data integration and comparison shopping. For these applications to further use the data embedded in HTML pages, automatic data extraction is necessary. Only when the data are extracted and organized in a structured way such as tables, they can be compared and aggregated. Hence, accurate data extraction is very important for these applications to perform correctly.

This paper focuses on the problem of automatically extract data records that are encoded in the query result pages generate by web databases. The goal of web database data extraction is to eliminate any unrelated information from the query result page, extract the query result records (referred to as QRRs in this paper) from the page, and align the extracted QRRs into a table such that the data value belong to the same characteristic are placed into the same table column.

The following two-step methods, called combine Tag and Value Similarity (CTVS), to extract the QRRs from a query result page p.

- Record extraction**:** Identifies the QRRs in p and involve two substeps: data region identification and the actual segmentation.
- Record alignment**:** Aligns the data values of the QRRs in p into a table so that values for the same attribute are aligned into the same table column.

CTVS precisely extracts and aligns the QRRs in query result pages if there are at least two records in the page. Compared with existing data extraction methods, CTVS improve data pulling out precision in three ways.

- New methods to imagine that the QRRs are existing contiguously in only one data region in a page. However, it is not best guess be true for many web databases where maintain in order separate the QRRs. We scan 10 websites to which the QRRs in the query result pages are non-contiguous, but indicate that noncontiguous data regions are rather familiar. Furthermore, 10 out 4 of websites have noncontiguous QRRS with the same parent in the page HTML tag tree. This problem is concentrate on two methods according to the outline of the QRRs and the major in series in the result HTML tag trees (i.e., DOM trees). a. An off-the-peg data region recognition method is likely to recognize the noncontiguous QRRs that have the similar parents according to their tag similarity. b. A combine method is proposed to join different data regions that contain the QRRs (with or without the same parent) into a single.
- In this we proposed to align the data values in the recognized QRRs, first pairwise then holistically. Together tag structure similarity and data value comparison are used in the pairwise alignment procedure i.e, first to join tag structure and data value similarity to achieve the alignment.
- A new nested-structure dealing out algorithm is future to handle any nested structure in

a) Compute the similarity computation method in the data region identification algorithm is recursively apply to the children of ni only if it does

b) Section the data region into data records using the record segmentation algorithm.

c) Suppose that the tag tree has n internal nodes and a node has a maximum of m children and a maximum tag string length of l. The time complexity of the data region identification algorithm is O(^{nm2l2}).

2.2 Record Segmentation

The Record Segmentation module then segments the identified data regions into data records according to the tag patterns in data regions. If only one tandem replicate is create in a data region, we assume that each frequent instance in the tandem

the QRRs after the holistic alignment i.e, CTVS uses both tag and data value similarity in sequence to get better nested structure processing exactness.

2. QRR EXTRACTIONS

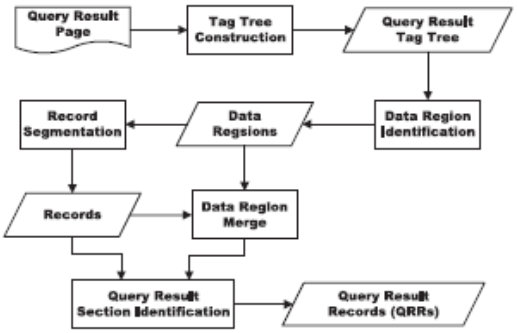

Fig.1. QRR extraction framework.

Fig.1 shows the framework for QRR extraction. In this aquery result page, the Tag Tree structure module first construct a tag tree for the page rooted in the <HTML>tag. Each node corresponds to a tag in the HTML page as well as its children are tags covered inside it. In every inner node n of the tag tree has a tag string tsn, which include the tags of n and all tags of n's descendants, and a tag path tpn, which include the tags from the root to n.

2.1 Data Region Identification

The Data Region Identification modules identify all likely data region, which usually contain dynamically generates data, top down starting from the root node. We first suppose that some child subtrees of the same parent node form alike data records, which assemble a data region. Thus, we propose a new process applied to more web databases. The data region identification algorithm is apply to a node n and recursively to its children $_{ni}$, i = 1 . . .m as follows:

not have any alike siblings. Several data regions may be recognized in this.

replicate corresponds to a record. If many tandem replicate are found in a data region, we need to select one to indicate the record. The two heuristics are used for the tandem replicate collection

a) If there is supplementary information, which corresponds to nodes between record instances, within a data region, the tandem replicate that stops at the supplementary information is the correct tandem replicate since information usually is not inserted into the middle of a record.

b) The optical gap between two records in a data region is usually larger than any optical gap within a record. Hence, the tandem replicate that satisfies this restriction is selected.

c) If the earlier two heuristics cannot be used, we select the tandem replicate that start the data region.

2.3 Data Region Merge

Given the segmented data records, the Data Region Merge module merges the data regions contain similar records. If any two data regions, we treat them as similar if the segmented records they contain are similar. The similarity between any two records from two data regions is measured by the resemblance of their tag strings. The resemblance between two data regions is calculated as the average record similarity. Two data regions can be merged into a merged data region if the records in the two data regions have an average similarity greater or equal to 0.6. Given, that two data regions d1 and d2 with n1 and n2 records and most record tag string length of l1 and l2, Respectively, the time complexity of the data region merges algorithm is O(n1n2l1l2).

2.4 Query Result Section Identification

The Query Result Section Identification module select one of the merged data regions as the one that contains the QRRs, there may still be multiple data regions in a query result page. Three heuristics are used to recognize this data region, called the query result section.

a) The query result section generally occupies a large space in the query result page. For each data region d, an area weight, which is calculated as d's area divided by the largest area of all identified data regions, is assigned for d.

b) The query result section is generally located at the center of the query result page. For each data region d, a center distance is considered among its center and the center of the page, and a center distance weight, which is calculated as the smallest center distance among all identified regions divided by d's center distance, is assigned for d. If a merged data region d contains multiple regions d1,. . . . . ,dn to be found in dissimilar parts of the page, then first find the region di that has the largest space in the query result page in the middle of d1,. . . , dn and assume that the center distance weight of di is the center distance weight of d.

c) Each QRR generally contains more raw data strings than the raw data strings in other sections. For each data region d, a value weight, which is calculate as the average number of raw data strings in the records of d divided by the largest average number of data values in all recognized regions, is assign for d. All the above three weights are summed and the data region that has the biggest summed weight is selected as the query result section. Records in this data region are supposed to be QRRs.

A restriction of this approach is that if a query result page has more than one data region that contains query result records and the records in the different data regions, then we will choose only one of the data regions and discard the others.

3. QRR ALIGNMENT

QRR alignment is performing by novel three-step data alignment methods that combine tag and value similarity.

3.1 Pairwise QRR Alignment

The pairwise QRR alignment aligns the data values in a pair of QRRs to present the proof for how the data values must be aligned among all QRRs. A pairwise alignment of r1 and r2 is composed of a set of data value alignments, each of which assumes that the corresponding data values from r1 and r2 belong to the same attribute.

Every QRR includes two kinds of information: the text string for the ith value and the tag path for the ith value. Throughout the pairwise alignment, we involve that the data value alignments must suit the following three constraints:

a) Same record path constraint: The record path of a data value f comprises the tag from the root of the record to the node that contains f in the tag tree of the query result page. Each pair of corresponding values have the similar tag path. Hence, if f1i has an altered tag path with f2j, then sij is assigned a small negative value to prevent the pair of values from being aligned.

b) Unique constraint: Each data value can be aligned for the most part one data value from the other QRR.

c) No cross alignment constraint: If f1i is matched to f2j, then there should be no data value alignment between f1k and f2l such that $k < i$ and $l > j$ or $k > i$ and $l < j$.

Based on these constraints, a dynamic programming algorithm aligns the two records. The similarity is 0 if one of the QRRs is empty or else, if f1i and f2j have the same tag path, then just one of the following three data value alignments is possible.

1. The first (i - 1) values of r1 can be aligned with the first (j - 1) values of r2 plus the data value alignment between f1i and f2j, which has the summing similarity score L(i-1)(j-1) + sij.

2. f1i can be ignored and the first (i - 1) values of r1 can be aligned with the first j values of r2, which has the summing similarity score L(i-1)j.
3. f2j can be ignored and the first i values of r1 can be aligned with the first j - 1 values of r2, which has the summing similarity score Li(j-1).
The alignment with the largest summing similarity score among these three alternatives is chosen. That is

$$Lij = \max(L_{(i-1)(j-1)} + sij, L(i-1)j, Li(j-1)) \quad (1)$$

With this dynamic programming method, the time complexity of the pairwise alignment algorithm is O(l1l2) where l1 and l2 are the number of data values in the two QRRs. Hence, given a data region with n records, the time complexity of the pairwise alignment algorithm is O(n2l2)in which l is the largest number of data values in a record.

3.1.1 Data Value Similarity Calculation:

Given two data values f1 and f2 from dissimilar QRRs, we need their similarity, s12, to be a real value in [0, 1]. The data value similarity is calculated according to the data type tree shown in Fig. 2.

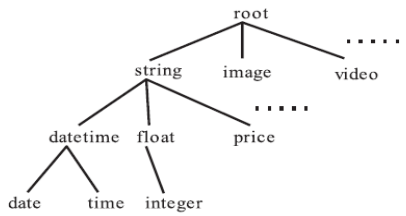

Fig. 2. Data type tree

Each child node is a subset of its parent node. For example, the "string" type includes several children data types, which are frequent on the web such as "datetime," "float," and "price." The highest depth of the data type tree is 4. We will transfer to a nonstring data type as a specific data type.
Given two data values f1 and f2, we first moderator data types and then fit them as extremely as possible into the nodes n1 and n2 of the data type tree. For example, given a string "567," we will put it in node "integer." The similarity s12 between two data values f1 and f2 with data type nodes n1 and n2 is defined as nonstring data type as a specific data type.

$$s12 = \begin{cases} 0.5 & n1 = p(n2) \ \& \ n1 \neq \text{String} \\ & \text{OR} \\ & n2 = p(n1) \ \& \ n2 \neq \text{String} \\ 1 & n1 = n2 \neq \text{String} \\ \text{cosine similarity} & n1 = n2 = \text{String} \\ 0 & \text{otherwise,} \end{cases}$$

where p(ni) refers to the parent node of ni in the data type tree. The similarity between data values f1 and f2 is
. 0.5, if they belong to different specific data types that have a common parent.
. 1, if they belong to the same specific data type.
. String cosine similarity of f1 and f2, if both f1 and f2 belong to the string data type.
. 0 otherwise, which occurs when one of f1 and f2 belongs to the string data type and the other one belongs to a specific data type, or f1 and f2 belong to different specific data types without any direct parent.
As Table 1 shows, data values with the same data type usually have larger similarity.

TABLE 1
Example of Data Value Similarity

f_1 / n_1	f_2 / n_2	s_{12}
23/int	876/int	1
Jul 1, 2007/date	2005.12.3, 1:30/datetime	0.5
$32.5/price	$18/price	1
Harry Potter and the Goblet of Fire/string	Harry Potter and the Prisoner of Azkaban/string	0.714
Harry Potter and the Goblet of Fire/string	876/int	0

3.2 Holistic Alignment

Holistic alignment align the data values in all the QRRs and this step of holistic alignment performs the alignment worldwide with all QRRs to create a table in which all data values of the same attribute are aligned in the same table column. Thus, holistic alignment problem is equal to that of finding connected components in an undirected graph. Each connected component of the graph represents a table column inside which the linked data values from different records are aligned vertically. We need to consider two application constraints that are specific to our holistic alignment problem.

a) Vertices from the same record are not allowed to be included in the same connected component as they are considered to come from two different attributes of the record. If two vertices from the same record breach this constraint, a path must exist between the two, which we call a breach path.

b) Connected components are not allowed to intersect each other. If C1 and C2 are two connected components, then vertices in C1 should be either all on the left side of C2 or all on the right side of C2, and vice versa.

So, we design a 3-step for the holistic alignment problem. First, we traverse the graph once by a depth-first search to discover the preliminary connected components in the graph (the Traverse and Visit functions). Throughout the traversal, a color array is use to indicate whether every vertex has been visited or not (WHITE for

unvisited, GRAY for under processing, and BLACK for processing finished). In the Visit function, when a new vertex is encounter, we add it into the current connected component. Second, at the same time we also mark those components containing breach paths. If a connected component is start containing breach paths, the BreakBreachPath function is called to break it by remove the edges with the smallest sum of pairwise similarity scores. Given two recognized nodes vi and vj from the same record, the problem of breaking a breach path is accurately the max-flow/min-cut problem in which vi and vj are the source and sink nodes, respectively.

Third, we traverse the components contain breach paths to eliminate some edges so as to break the breach paths (i.e., enforce the first constraint). Finally, we use a divide-and-conquer method to recognize and split up the intersecting components to inflict the second constraint.

3.3 Nested Structure Processing

This identifies the nested structures that exist in the QRRs. If a QRR contains a nested structure such that an attribute has multiple values, then several of the values might not be aligned to any other values. Hence nested structure giving out identifies the data values of a QRR that are generated by nested structures. Relying only on HTML tags to recognize nested structures, as is done by approximately all existing methods, may incorrectly recognize a plain structure as a nested one. To overcome this problem, CTVS uses both the HTML tags and the data values to identify the nested structures.

Given an aligned table, a nested column comprises at least two ordered sets signifying the data values that are generated by repetitive parts in the template. A nested column set C is comprised of a set of nested columns. For example, in the nested column set {{<1,2>,<3, 4>,<5, 6>},{<7>, <8>}}, two nested columns are identified. The first nested column denotes that the first to sixth columns of the table are generated from a repetitive part three time. The first, third, and fifth columns are generated from one attribute and the second, fourth, and sixth columns are generated from another attribute. Similarly, the seventh and eighth columns are generated by another repetitive part of the template. Considering the two QRRs in The nested column set C in this table is {{<3, 4, 5, 6,7>,<8, 9, 10, 11, 12>}}.

Given columns cp in a holistic alignment and a similarity threshold Snest as input, the procedure nested decides, using the similarities of the data values in cp, whether cp contains a repetitive tag pattern that is formed by a nested structure. We assume that two columns are generated by the same attribute if there is a large data value similarity between these two columns. Given a column c1, which contains m data values, we define the intracolumn similarity simintra to be the average data value similarity within each column in c1.

$$\text{Simintra} = 2\sum_{j=1}^{m-1}\sum_{i=j+1}^{m} sij \;/\; m(m-1) \qquad (2)$$

In (2), sij is the data value similarity among the ith and jth data values of c1. For cp, its intracolumn similarity is the average of the intracolumn similarity of all columns in cp .

For two columns c1 and c2, which have m and n data values, respectively, the intercolumn similarity siminter is defined to be the average data value similarity of every pair of data values in c1 and c2 .

$$\text{Siminter} = \sum_{j=1}^{n}\sum_{i=1}^{m} \text{sij} / mn \qquad (3)$$

In (3), sij is the data value similarity among the ith data values of c1 and jth data values of c2 using the data value similarity calculation described in Section 3.1.1.

After siminter and simintra are calculated for identified columns cp, if siminter=simintra > Snest, where Snest is a threshold that is set to 0.5, cp is assumed to be a nested column set, which means that the data values in it are generated from a nested structure.

Given data columns cp and the nested column set C as input, the method add_nested_column adds the nested columns cp to C. Then ci in C is replaced with cp .Otherwise, cp is simply added as a new element into C .

Given n records with a maximum of m data values and a maximum tag string length of l, the time complexity of the nested structure processing algorithm is O(nl2m2). For each record, at most O(l2) time is needed to identify repetitive tag patterns; if a pattern is found, at most O(m2) time is required to calculate the intra-/intercolumn similarity.

Compare the nested structure processing methods in DeLa [29] and NET [20], the nested structure processing technique in CTVS has the following advantages.

a) CTVS processes the nested structures after the data records are aligned rather than before as is the case in DeLa and NET. The nested structure before the records are aligned makes them weak to optional attributes that makes the tag structure irregular. This difficulty does not arise in CTVS.

b) In CTVS the data value similarity information efficiently prevents a flat structure from being identified as a nested structure. It shares similar tag structures, a flat structure by several columns having the same tag structure, may be wrongly identified as a nested structure one can have serious consequences in DeLa and NET. DeLa condenses all the values into one parent value and then aligns them to other records, which makes

the alignment much more complex. If NET wrongly identifies a simple structure as a nested structure, it will create a new row in the table for each data value of the simple structure.

4. Experimental Results

The act of the data extraction methods is compare in three different ways. General data set evaluation present the act on the first three data sets, which display a variety of properties and have been used in earlier work by others. The other two evaluations focus on exact properties of the query result pages. Noncontiguous QRR evaluation compares the act for query result pages in which the QRRs are contiguous and noncontiguous. Nested-structure evaluations compare the performance for query result pages with and without a nested structure.

5. Related Work

In wrapper induction, extraction systems are copied based on inductive learning. This not scalable to a large number of web databases. Hence, the wrapper induction approach involves two additional difficult problems: monitoring changes in a page's format and maintaining a wrapper when a page's format changes. To conquer the problems of wrapper induction, some unsupervised learning methods, such as, IEPAD, ExAlg , DeLa , have been planned to repeatedly extract the data from the query result pages. To conquer these shortcomings, methods such as ViPER and ViNTs make use of extra information in the query result pages.

All the works make use of only the information in the query result pages to execute the data extraction. There are works that make use of extra information, specifically ontologies, to aid in the data extraction. While these approaches can overcome some of the limitations of CTVS (e.g., that a query result page should contain at least two QRRs) and can get high accuracy, they need the availability of extra resources to construct an ontology as well as the additional step of actually constructing the ontology.

6. CONCLUSIONS AND FUTURE DIRECTIONS

We presented a new data extraction method, CTVS, to repeatedly extract QRRs from a query result page. CTVS employs two steps i.e. the first steps identify and segment the QRRs. These develop on alive technique by allowing the QRRs in a data region to be non-contiguous. The second step aligns the data values between the QRRs. A new alignment method is proposed in which the alignment is performing in three successive steps: pairwise alignment, holistic alignment, and nested structure processing.

Although CTVS has been shown to be a correct data extraction method, it still suffers from some restrictions. First, it requires at least two QRRs in the query result page. Second, any optional attribute that appear as the start node in a data region will be treat as auxiliary information. Third, CTVS mostly depends on tag structures to find out data values. Therefore, CTVS does not hold the case where multiple data values from more than one attribute are clustered inside one leaf node of the tag tree, as well as the case where one data value of a single element spans multiple leaf nodes.

REFERENCES

[1] B. Liu and Y. Zhai, "NET - A System for ExtractingWeb Data from Flat and Nested Data Records," Proc. Sixth Int'l Conf. Web Information Systems Eng., pp. 487-495, 2005.

[2] J. Wang and F.H. Lochovsky, "Data Extraction and Label Assignment for Web Databases," Proc. 12th World Wide Web Conf., pp. 187-196, 2003.

[3] K. Simon and G. Lausen, "ViPER: Augmenting Automatic Information Extraction with Visual Perceptions," Proc. 14th ACM Int'l Conf. Information and Knowledge Management, pp. 381-388, 2005.

[4] Y. Zhai and B. Liu, "Structured Data Extraction from the Web Based on Partial Tree Alignment," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 12, pp. 1614-1628, Dec. 2006.

[5] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu, "Fully Automatic Wrapper Generation for Search Engines," Proc. 14th World Wide Web Conf., pp. 66-75, 2005.

[6] M.K. Bergman, "The Deep Web: Surfacing Hidden Value," White Paper, BrightPlanet Corporation, http://www.brightplanet. com/ resources/details/deepweb.html, 2001.

[7] K.C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang, "Structured Databases on the Web: Observations and Implications," SIGMOD Record, vol. 33, no. 3, pp. 61-70, 2004.

[8] C.H. Chang and S.C. Lui, "IEPAD: Information Extraction Based on Pattern Discovery," Proc. 10th World Wide Web Conf., pp. 681- 688, 2001.

[9] A. Arasu and H. Garcia-Molina, "Extracting Structured Data from Web Pages," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 337-348, 2003.

[10] H. Snoussi, L. Magnin, and J.-Y. Nie, "Heterogeneous Web Data Extraction Using Ontologies," Proc. Fifth Int'l Conf. Agent-Oriented Information Systems, pp. 99-110, 2001.

[11] W. Su, J. Wang, and F.H. Lochovsky, "ODE: Ontology-Assisted Data Extraction," ACM Trans. Database Systems, vol. 34, no. 2, article 12, p. 35, 2009.

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