International Journal of Computer Applications (0975 - 8887)
Volume 60 - No. 5, December 2012

Feature-based Clustering of Web Data Sources

Alsayed Algergawy
Computer Engineering Department
Tanta University
Tanta, Egypt

ABSTRACT

The proliferation of web data sources increasingly demands the integration of these sources. To facilitate the integration process, a pre-analysis step is required to classify and group data sources into their correct domains. In this paper, we propose a feature-based clustering approach for clustering web data sources without any human intervention and based only on features extracted from the source schemas. In particular, we make use of both linguistic and structural schema features. We experimentally demonstrate the effectiveness of the proposed approach in terms of both the clustering quality and runtime.

General Terms:

Algorithms, Design, Experiments

Keywords:

Web data source, Data integration, Clustering, Performance.

1. INTRODUCTION

The number of structured data sources on the web is rapidly increasing, e.g. the study in [5] estimated about 450,000 such sources. Common examples of structured web sources are websites that support access to databases through web forms. Such databases hidden behind web forms constitute the so-called *deep web*. A recent study by Google [6] reports about 10 million web forms. A common characteristic of these structured sources is that they exhibit a large degree of semantic heterogeneity resulting in largely different forms even within the same domain. This raises the need to provide unified form-based access to different sources of a domain, e.g. similar to some meta search engines [9].
To this end, a few approaches have been proposed. A first approach is to model structured data sources as *documents* and then apply *keyword search* on them. This approach has two main disadvantages. Treating deep web data as documents completely ignores the structure feature of these data sources. Furthermore, applying keyword search results in a huge number of false positive candidates which dramatically reduces the quality of the search process. Other approaches that consider structural features are to use data integration. At web scale, the massive number of data sources makes even semi-automatic data integration techniques impractical. As well as existing fully automatic data integration techniques assume that the data sources are homogenous, i.e. belonging to the same domain. Therefore, a pre-process step is required to group and cluster similar data sources before applying the integration process.
Only a few approaches have been proposed to group and classify web data sources [7, 2, 1]. [2, 1] uses only element properties that are represented in Web forms in the clustering process. [7] proposes an approach to cluster Web data sources based on their schemas using only schema elements'names. It completely ignores the structural feature of these schemas.
In this paper, we propose a new approach for clustering web data sources based on their schema (metadata) features. In particular, the approach makes use of both linguistic and structural features of schema elements. We construct vectors per schema from these features and use the vectors to compute the similarity between schemas. A hierarchical clustering algorithm is proposed to group together similar schemas of the same domain. An experimental evaluation validates the effectiveness and performance of the proposed approach.
The rest of the paper is organized as follows. We introduce basic definitions in Section 2. The feature-based clustering approach is described in Section 3. Section 4 reports the experimental results. We conclude in Section 5.

2. PRELIMINARIES

In this section we present definitions and basic concepts that be used through the paper.

2.1 Schema graph

In order to make the proposed approach generic, we represent structured web data such as web forms or the schemas of the underlying databases (e.g., XML schemas and ontologies) as labeled directed acyclic graphs, called *schema graphs* (SG).
Definition 1. A schema graph is a rooted node-labeled directed acyclic graph. It is represented as a 3-tuple (V, E, Lab_v), where:

—$V = \{r, v_2, ..., v_n\}$ is a finite set of nodes, each of them is uniquely identified by an object identifier (OID), where r is the schema graph root node.

—$E = \{(v_i, v_j)|v_i, v_j \in V\}$ is a finite set of edges.

—Lab_v is a finite set of node labels. These labels are strings for describing the properties of the element and attribute nodes, such as *name* and *data type*.

Figs. 1 and 2 show schema graphs representation of two web data sources taken from [3]. *DeptDB* represents information about departments with their employees and grants, as well as the projects for which grants are awarded. *OrgDB* is a variation of *deptDB*, where employees and funds are now grouped by organizations. Furthermore, *OrgDB* includes additional information not present in *DeptDB* (e.g., organizations have addresses and employees have phones.). The figures show that each node is associated with the node name and the node identifier. For example, in Fig. 1, the node v_1 has the name *DeptDB*.

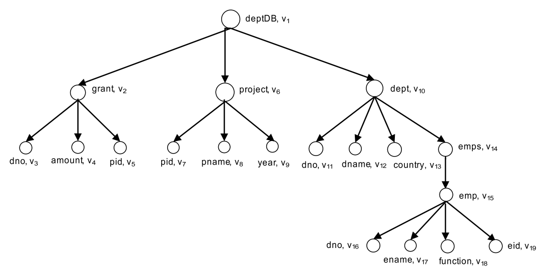

Fig. 1: Schema graph, deptDB

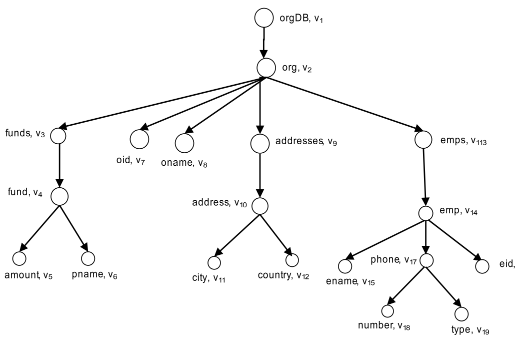

Fig. 2: Schema graph, orgDB

2.2 Schema feature vectors

Each element (node) of a schema graph has its own internal features, such as name, type, and constraint as well as structural features that represent the element's position in the schema graph. It is known that the element name is the most useful internal feature for schema elements while no single structural feature is commonly most significant. Therefore, in our implementation, we consider both the element name and a set of the element structural features for the clustering process.

We therefore construct from each schema graph two feature vectors, called *TermVector* and *StrVector*. *TermVector* is used to capture the linguistic properties of each schema. It is constructed by the names of schema elements. Each name is extracted from the schema and normalized by tokenizing it into a set of tokens. Distinct tokens with length higher than two are sorted into *TermVector*. For example, term vectors of the two data sources shown in Figs. 1 and 2 are $TV_1\{$*pname, dept, dno, pid, ename, eid, emp, grant, year, emps, amount, function, country, dname, project*$\}$ and $TV_2\{$*eid, ename, emp, emps, amount, type, oid, phones, address, country, org, city, funds, pname, fund, addresses, phone, number, oname*$\}$, respectively.

StrVector is used to represent the structural features of the schema graph. We utilize a combination of a set of ten structural element features, including the number of roots, the number of levels, the total number of elements, the total number of paths, etc. For example, term vectors of the two data sources shown in Figs. 1 and 2 are SV_1=*[1.0, 5.0, 0.0, 19.0, 6.0, 13.0, 19.0, 6.0, 13.0, 18.0]* and SV_2=*[1.0, 7.0, 0.0, 20.0, 10.0, 10.0, 20.0, 10.0, 10.0, 19.0]*.

3. FEATURE-BASED CLUSTERING

Given a set of heterogeneous web data sources each represented by a schema graph, the problem is to group similar sources into domains such that the intra-domain similarity is sufficiently high and the inter-domain similarity is sufficiently small. In the following we present how to compute the similarity between data sources and then introduce the clustering algorithm for grouping these data sources.

3.1 Inter-schema similarity computation

In order to group similar data sources into domains, we compute the similarity between every schema pair by using the two feature vectors per schema. Given a set of web data sources with schema graphs $SG_1, SG_2, ..., SG_n$, we construct a $n \times n$ schema similarity matrix, *SchMat*, to assess the similarity between every schema pair. To take both the linguistic and structural schema features into account, we use the following formula to determine the similarity between schemas i and j:

$$Sim(SG_i, SG_j) = w \times termVectorSim(TV_i, TV_j) + (1-w) \times strVectorSim(SV_i, SV_j) \quad (1)$$

where TV_i/SV_i and TV_j/SV_j are the term/structure vectors of schemas SG_i and SG_j, respectively. $termVectorSim$ is the similarity measure to compute the term vector similarity. To this end, we make use of the Jaccard coefficient given as

$$termVectorSim(TV_i, TV_j) = \frac{|TV_i \cap TV_j|}{|TV_i \cup TV_j|} \quad (2)$$

$strVectorSim$ is the similarity measure used to compute the structural similarity. We use the cosine similarity given as

$$strVectorSim(SV_i, SV_j) = \frac{\sum_{k=1}^{m} SV_{ik} \times SV_{jk}}{\sqrt{\sum_{k=1}^{m} SV_i^2 \times \sum_{k=1}^{m} SV_j^2}} \quad (3)$$

where m is the number of items in the structural vector. Finally, w is a weight used to quantify the importance of each similarity measure.

Example 1. *The term and structure similarities between the two data sources shown in Figs. 1 and 2 are* $termVectorSim(deptDB, orgDB) = 0.29$ *and* $strVectorSim(deptDB, orgDB) = 0.5$, *respectively. Setting the weight w to 0.5, the inter-schema similarity between the two data sources is 0.39.*

3.2 The clustering algorithm

Our goal is to group similar web data sources into domains. Clustering is a useful technique for grouping schemas such that schemas within the same domain are similar while schemas in different domains are dissimilar. The clustering algorithm presented in this paper is an agglomerative hierarchical algorithm mainly extended from the SCAN approach [11]. The hierarchical clustering algorithm is appropriate for this clustering task since in general we do not know the number of clusters in advance. The algorithm produces a tree called *dendrogram* representing the hierarchy of clusters in a bottom-up fashion. As shown in Algorithm 1, the proposed clustering algorithm proceeds in four steps as follows:

—**Preparation.** The algorithm accepts a set of web data sources (schemas), S, to be clustered and prepares it for the next stages. The stage starts by initializing the output set of clusters (*ClusterSet*) and the cluster hierarchy (*Dendro*), $line$ 1. Then, the algorithm computes the inter-schema similarities (*SchMat*) in advance to avoid recomputing them multiple times during clustering, $line$ 2. The computation is based on Eq. 1.

—**Cluster initialization.** It constructs the bottom level of the cluster hierarchy. Each schema represents its own cluster resulting into n clusters in the cluster set (*ClusterSet*),

$lines\, 3\, to\, 6$. Once getting the initial cluster set, the bottom level of the hierarchy is added to the dendrogram, $line\, 7$.

Algorithm 1 Clustering algorithm

```
Require: A set of schemas, $S = \{S_1, S_2, ..., S_n\}$
Ensure: A set of clusters, $Clust\_Set$
    {// Step 1: Preparation}
 1: $ClusterSet \Leftarrow \phi$, $Dendro \Leftarrow \phi$;
 2: $SchMat[][] \Leftarrow computeInterSchema(S)$
    {// Step 2: Cluster initialization}
 3: for $S_i \in S$ do
 4:     $Cluster\, C \Leftarrow new\, Cluster(S_i)$;
 5:     $ClusterSet.add(C)$;
 6: end for
 7: $Dendro.addLevel(ClusterSet)$;
    {// Step 3: Cluster hierarchy construction}
 8: $dist \Leftarrow 1$;
 9: while $ClusterSet.size() > 1$ do
10:     $k \Leftarrow ClusterSet.size()$;
11:     $ClusterSet \Leftarrow mergeCluster(dist)$
12:     if $k > ClusterSet.size()$ then
13:         $Dendro.addLevel(ClusterSet)$;
14:         $computeIntraSim(ClusterSet)$;
15:         $k \Leftarrow ClusterSet.size()$;
16:     end if
17:     if $noMoreMerge()$ then
18:         $break$;
19:     end if
20:     $dist \Leftarrow dist + \delta$;
21: end while
    {// Step 4: Best cluster set selection}
22: return $Dendro.getBestCluster(BestLevel)$;
```

—**Cluster hierarchy construction.** This is the main stage of the clustering algorithm and is dedicated to constructing the cluster hierarchy. It first initializes the distance between levels of hierarchy with 1, $line\, 8$. The algorithm iteratively merges clusters at a certain level until either the number of clusters reaches 1 or there is no possibility to merge more clusters. We keep the current size of the cluster set in variable k, $lines\, 10\&\, 15$. If the number of clusters after merging is changed, $line\, 12$, the new cluster set is added to the cluster hierarchy at the specified level. Furthermore, as we will explain later, the intra-cluster similarity is computed and the k value is updated. After that the algorithm checks if there is a possibility to further merge clusters and finally updates the distance for the current hierarchy level.

Having two clusters C_1 and C_2 containing k_1 and k_2 elements respectively, the similarity between them can be expressed as the average inter-schema similarity. It can be represented as follows:

$$Sim(C1, C2) = \frac{\sum_{i=1}^{k_1} \sum_{j=1}^{k_2} Sim(S_{1i}, S_{2j})}{k_1 + k_2} \quad (4)$$

where $sim(S_{1i}, S_{2j})$ is the inter-schema similarity between $S_{1i} \in C_1$ and $S_{2j} \in C_2$ computed by Eq.0**??**. Having this similarity between every cluster pair, a condition is required to decide if elements in the cluster pair should be merged. This condition has to reflect the level of the cluster hierarchy at which elements come together. Therefore, the introduced distance variable is used ($dist$, $line\, 8$). Elements of every cluster pair are combined when the similarity between the two clusters exceeds the predefined level similarity threshold ($1/dist$). The value of the level distance is then updated to reflect the nature of the next level ($line\, 20$, Algorithm 1).

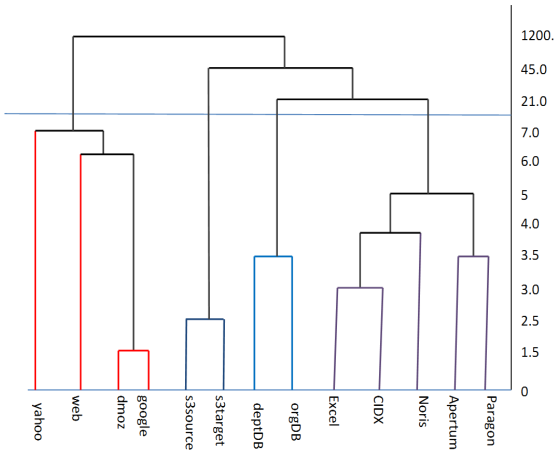

Fig. 3: Using name feature.

—**Best cluster set selection.** The task of the final stage is to select the best cluster set. Each level in the dendrogram is associated with a value that represents the average value of intra-cluster similarities of clusters at that level. The algorithm returns the cluster set at the level with the best value, $line\, 22$.

4. EXPERIMENTAL EVALUATION

In order to evaluate the performance of the proposed approach, we conducted a set of experiments utilizing real-world ontologies of different sizes representing heterogeneous web data sources. We ran all our experiments on a 2.66 GHz Intel(R) Xeon(R) processor with 4 GB RAM running Windows 7. We implemented the approach in Java.

Table 1. : Data set specification

Series	Tested sources	No. data sources	min./max. No. of elements
1	PO (XDR)	5	27/614
2	Webdirectory (OWL)	4	418/3234
3	Spicy (XSD)	4	20/125

4.1 Data set

Table 1 shows the characteristics of the test schemas from different domains and represented in different formats. Series 1 contains five XML schemas for purchase orders (PO) taken from [4]. Series 2 includes five ontologies from the Web directory domain [8]. Series 3 contains four XML schemas from [10] belonging to two different domains. More details about data sets in Table 1 can be found in [4, 8].

4.2 Experimental Results

We present results for three sets of experiments. The first set is to study the effect of the term vector feature on the clustering process. The effect of the structural feature is then investigated in the second set. Finally, the combination of both features is studied. Results are shown in Figs. 3, 4, and 5. Each figure represents a cluster hierarchy (dendrogram), where the horizontal axis represents data sources that to be clustered and the vertical axis represents the dendrogram levels at different distances. As mentioned in Algorithm 1, the distance ($dist$) is set to an initial value (in the set of experiments $dist = 0$) and it then increases by a step of 0.5 ($\delta = 0.5$). When the similarity between two data sources (or two set of clusters) is higher than the current distance, the two sources (sources of the two clusters) are then merged to form a new level in the cluster hierarchy.

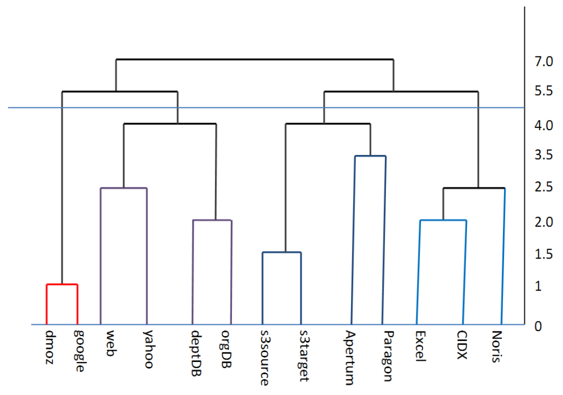

Fig. 4: Using structure feature.

Fig. 3 represents the cluster hierarchy using only the term vector feature. It should be noted that the algorithm needs a distance of 1200.5 to construct the dendrogram. According to Step 4 in Algorithm 1, we obtained the best cluster solution at level 9 ($dist = 7.0$). The cluster solution consists of four sets of clusters, each representing a specific domain. Fig. 4 shows the cluster hierarchy using only the structural vector feature. It needs a distance of 7.0 to build the dendrogram. The best cluster solution occurs at level 7 ($dist = 4.0$) and consists of four sets of clusters. The cluster hierarchy produced by using both the term vector and structural vector features is illustrated in Fig. 5. To construct the dendrogram, it requires a distance of 22.5. The best cluster solution is reached at level 9 ($dist = 6.05$).
To validate the performance of the cluster solution, we used both the quality of cluster solution and the run time evaluation. Standard measures, such as Precision, Recall, and F-measure are used as criteria for the quality, and the number of iterations required to construct the dendrogram is used as a criterion for run time. Results are reported in Table 2. The table shows that using only the structural vector feature is not sufficient to correctly cluster web data sources into their corresponding domains. This can be explained as follows: First, different Web data sources are structured and engineered by different people. So, there is a large possibility that data sources from different domains are modeled by the same structure. Furthermore, in our implementation, we use some ad-hoc combination of several structural features that cannot correctly quantify the similarity between all sources. However, the table indicates that using the structural feature is the fastest method to construct the dendrogram.
Table 2 also shows that both using the term vector and using both vectors produce a perfect cluster solution (F-measure=1). This can be explained as data sources from the same domain tend to used common terms, which makes the term vector adequate to correctly quantify the similarity between data sources. It should be noted that using only the term vector requires a large number of iterations to build the dendrogram and with the combined consideration of the structural vectors speed up the clustering. The table indicates that using only the term vector requires 2400 iterations, which reduces to 45 when the structure vector is combined with the term vector.

Table 2. : Performance result

	Quality			Run time	
	P	R	F-meas.	$dist_{max}$	No. of iterations
Term vector	1	1	1	1200.5	2400
Str. vector	0.775	0.75	0.76	7.0	14
Both	1	1	1	22.5	45

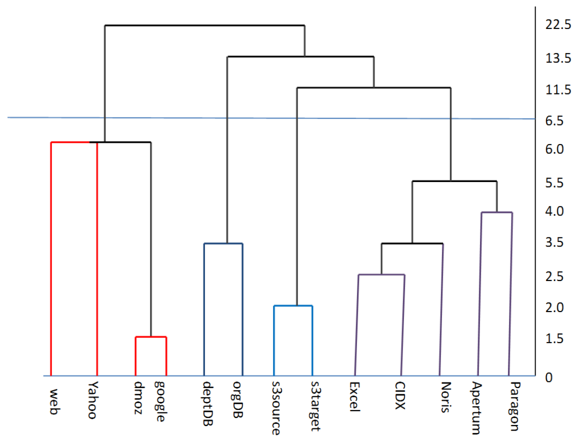

Fig. 5: Using both features.

5. CONCLUSIONS

Data sources on the web are proliferating and the need to develop high performance techniques to manage them is crucial. We proposed a feature-based clustering approach to group web data sources into their corresponding domains. The approach is generic, can deal with different data models and does not depend on human intervention. We carried out a set of experiments to validate the performance of the approach. Results show that combining both the term vector and the structural vector outperforms the other settings. Further work will investigate the extension of the approach to integrate more features such as semantic information to further improve the approach.

6. REFERENCES

[1] L. Barbosa and J. Freire. Combining classifiers to identify online databases. In *WWW*, 2007.

[2] L. Barbosa, J. Freire, and A. S. da Silva. Organizing hidden-web databases by clustering visible web documents. In *ICDE*, pages 326–335, 2007.

[3] L. Chiticariu, M. A. Hernndez, P. G. Kolaitis, and L. Popa. Semi-automatic schema integration in Clio. In *VLDB'07*, pages 1326–1329, 2007.

[4] H. H. Do and E. Rahm. Matching large schemas: Approaches and evaluation. *Information Systems*, 32(6):857–885, 2007.

[5] T. M. Ghanem and W. G. Aref. Databases deepen the web. *Computer*, 37(1):116–117, 2004.

[6] J. Madhavan, S. R. Jeffery, S. Cohen, X. L. Dong, D. Ko, C. Yu, and A. Halevy. Web-scale data integration: You can only afford to pay as you go. In *CIDR*, pages 342–350, 2007.

[7] H. A. Mahmoud and A. Aboulnaga. Schema clustering and retrieval for multi-domain pay-as-you-go data integration systems. In *SIGMOD*, 2010.

[8] S. Massmann and E. Rahm. Evaluating instance-based matching of web directories. In *11th Workshop on Web and Databases (WebDB)*, 2008.

[9] W. Meng and C. T. Yu. *Advanced Metasearch Engine Technology*. Morgan & Claypool Publishers, 2010.

[10] E. Peukert, S. Massmann, and K. Konig. Comparing similarity combination methods for schema matching. In *GI-Workshop*, pages 692–701, 2010.

[11] N. Yuruk, M. Mete, X. Xu, and T. A. J. Schweiger. AHSCAN: Agglomerative hierarchical structural clustering algorithm for networks. In *ASONAḾ09*.