Combining Multiple Features for Web Data Sources Clustering

Alsayed Algergawy
Department of Computer Science,
Otto-von-Guericke University,
39106 Magdeburg, Germany
Department of Computer Engineering,
Tanta University, Egypt

Gunter Saake
Department of Computer Science
Otto-von-Guericke University
39106 Magdeburg, Germany
Email: saake@iti.cs.uni-magdeburg.de

***Abstract*—The numbers of web data sources grow significantly, and as a sequence, crucial data management issues should be addressed. Clustering is one of the issues that many researchers have focused on. Clustering has been proposed to improve the information availability. To this end, 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 combine linguistic and structure features of each data source to enhance computation of schema similarity. We experimentally demonstrate the effectiveness of the proposed approach in terms of both the clustering quality and runtime.**

I. Introduction

The explosive growth of Web data sources has dramatically changed the way in which these data are managed and accessed. Thus, there is a growing need to propose and develop high-performance techniques for efficiently managing and analyzing extremely large collections of web data. One of the methods that many researchers have focused on is clustering, which groups similar web data according to their content, and structures. The clustering process of web data plays a crucial role in many data application domains, such as information retrieval, data integration, document classification, Web mining, and query processing [2]. It increases web data availability and improves content delivery on the web.

It has been known that the number of structured data sources on the web is rapidly increasing. A study, conducted in year 2000, estimated that the public information in deep web is 500 times larger than the surface web, with 7,500 Terabytes of data across 200,000 deep web sites [7]. A recent study by Google [13] reports about 10 million web forms. 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 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.

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 [14]. 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 [15], [6], [5]. [6], [5] uses only element properties that are represented in Web forms in the clustering process. [15] 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, we start by analyzing web data sources to extract linguistic and structure properties (features). Each schema feature is then represented as a vector, which can be used to compute the schema similarity. We combine both the linguistic and structure features in order to improve the schema similarity. A hierarchical clustering algorithm is proposed to group together similar schemas of the same domain. Finally, we conducted an intensive set of experiments to validate the effectiveness and performance of the proposed approach.

Our main contributions are the followings:

- proposing and developing a generic web data clustering approach that is able to cope with different data models, such as XML data, RDF, and ontologies.
- proposing a combining technique that combines multiple web data features, such as element name and structure.
- conducting a set of experiments to validate the performance of the proposed approach.

II. Related Work

In general, web information sources can be categorized into two classes: web user clustering and web schema/document

clustering [21]. The first class, web user clustering, is to form groups of users exhibiting similar browsing patterns [11]. However, web schema/document clustering is the grouping of web data sources according to either their content, structure, semantic or two of them, or all these features [2]. Web document clustering has been intensively addressed and a lot of approaches have been proposed. Some make use of the content of web documents, other approaches exploit the structure feature of documents [9], [1], and others utilize both [20]. A comprehensive coverage is provided in [2]. Contradictory, a few approaches have been proposed to address the web schema clustering problem [12], [17], [4], [15].

XClust proposes an approach of clustering the DTDs of XML data sources to effectively integrate them [12]. It represents DTD schemas as data trees. To compute the similarity between two data trees, it depends on the computation of similarity between tree nodes exploiting tree nodes' features. This approach works well when dealing with small web data sources, however, the similarity computation between individual elements is very expensive process when dealing with large data sources. Similarly, approaches in [4], [17] compute the similarity between XML schemas, represented as schema trees, based on the computed similarities between schema elements. However, both approaches develop some optimization techniques to accelerate the clustering process.

Similar to our approach is the technique proposed in [15]. This technique is proposed to cluster web data based on exploiting schema elements'name. It does not need to compute the similarity between every element pair, however, it completely ignores other schema'features, such as structural feature.

III. Preliminaries

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

A. *Schema graph*

Web data sources can be modeled using different languages and data models, such as XML, RDFs, and ontology. In order to make the proposed approach more generic, we represent structured web data such as the schemas of the underlying databases 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 [8]. *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.

B. *Schema feature vectors*

After representing each web data source as a schema graph, the next step is to extract schema features that can be used for computing the similarity. As given, each schema graph consists of a set of labeled nodes connecting by a set of edges. 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. In our implementation, we consider both the element name and a set of element structure features for the clustering process. We start by analyzing each schema graph and extracting its name and structure features. We then construct from a schema graph two feature vectors, called *Term Vector (TermVector)* and *Structure Vector (StrVector)*.

1) Term feature vector: 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 [3]. Furthermore, in the absence of data instance such as clustering web data based on their metadata, the element name is considered an important source of semantic information for similarity computation. Element names can be syntactically similar or semantically similar. Even it is desirable to consider both syntactic and semantic aspects of element names to compute the degree of similarity between element names, we only consider the syntactic aspect of element names.

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.

2) Structure feature vector: Web data sources are engineered by different organizations, even if they represent the same domain. As a result, several semantic and structural heterogeneities exist across these data sources. Utilizing only the element name feature to compute the similarity between web data sources becomes insufficient and other schema features should be exploited. To this end, we consider structure schema feature. In fact, there is no single structure feature that can be described as the most significant feature [3]. Given that the classical structure measures such as path and context are time consuming, which affects the process of web data clustering. We thus 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 to construct the structure feature vector *StrVector*.

IV. Feature-based Clustering

Given a set of web data sources (schemas) $S = \{S_1, S_2, ..., S_m\}$, each represented by a schema graph, the objective is to get a set of clusters $C = \{C_1, C_2, ..., C_k\}$,

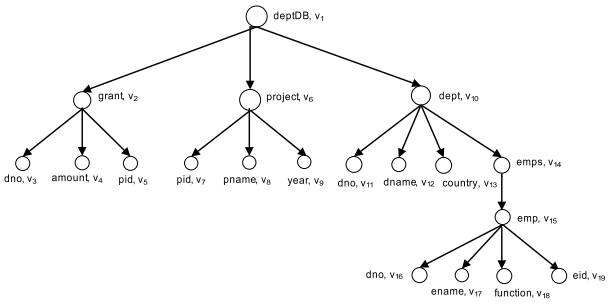

Fig. 1: Schema graph, deptDB

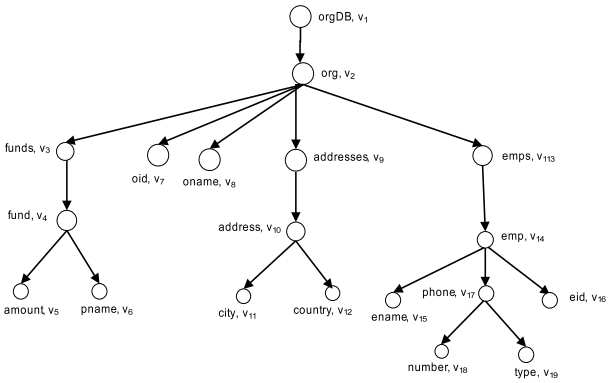

Fig. 2: Schema graph, orgDB

such that similar schemas belong to the same cluster, while dissimilar schemas are in different clusters. In this context, 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.

A. Inter-schema similarity computation

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. We observe that computing the similarity between web data sources involves multiple aspects (in our implementation we consider only name and structure features), which should be optimized simultaneously. Optimizing one aspect, for example the name similarity, will affect the other aspects. Hence, we need a compromise between these aspects, giving that semantic and structure heterogeneities among web data sources. To take both the linguistic and structural schema features into account, we reformulate the multi-objective problem into a single objective problem to determine the similarity between schemas i and j: as follows:

$$\begin{aligned} Sim(SG_i, SG_j) =& Combine(Sim_t(TV_i, TV_j), \\ & Sim_{st}(SV_i, SV_j)) \end{aligned} \quad (1)$$

where TV_i/SV_i and TV_j/SV_j are the term/structure vectors of schemas SG_i and SG_j, respectively, Sim_t is the similarity measure to compute the term vector similarity, Sim_{st} is the similarity measure used to compute the structural similarity, and $combine$ is a function used to combine the both similarity measures. To compute the similarity between extracted term vetcors, we make use of the Jaccard coefficient given as

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

Similarly, the cosine similarity measure is used to compute the similarity between extracted structure features, as follows:

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

where m is the number of items in the structural vector.

The arising question now is how to combine these similarity measures, such that the total similarity value is optimized. I.e., it has a maximum value when the two schema graphs, SG_i and SG_j are semantically and structural similar, while it should be minimum when they are dissimilar. As defined the problem as a multi-objective problem, there are many methods used to combine similarity measures. The most easy and perhaps most widely used method is the weighted-sum method [19]. Therefore, Equation 1 can be rewritten as follows:

$$\begin{aligned} Sim(SG_i, SG_j) =& w \times Sim_t(TV_i, TV_j) \\ & + (1 - w) \times Sim_{st}(SV_i, SV_j) \end{aligned} \quad (4)$$

where 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 $Sim_t(deptDB, orgDB) = 0.29$ and $Sim_{st}(deptDB, orgDB) = 0.5$, respectively. Setting the weight w to 0.5 and $\sigma = 0.7$ the inter-schema similarity between the two data sources is 0.39.

B. The clustering algorithm

Our goal is to group similar web data sources into domains. The clustering algorithm presented in this paper is an agglomerative hierarchical algorithm mainly extended from the SCAN approach [22]. 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, $line\,2$.
- **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 stage 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. 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:

$$Sim(C1, C2) = \frac{\sum_{i=1}^{k_1}\sum_{j=1}^{k_2} Sim(S_{1i}, S_{2j})}{k_1 + k_2} \quad (5)$$

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.1. 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).

- **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, $line\,22$.

V. Experimental Evaluation

In order to evaluate the performance of the proposed approach, we conducted a set of experiments utilizing real-world schemas and 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 I: Data set specification

Domain	Tested sources	No. data sources	min./max. No. of elements
1	PO (XDR)	20	27/1451
2	Webdirectory (OWL)	8	418/3234
3	Spicy (XSD)	4	20/125
4	OAEI benchmark (OWL)	11	46/126
5	Anatomy (OWL)	2	2746/3306
6	University (XSD)	17	8/25
7	Other (XSD)	11	10/120

A. Data set

Table I shows the characteristics of the test schemas and ontologies from different domains and represented in different formats. For examle, data set in Domain 1 contains ten XML schemas for purchase orders (PO) taken from [10]. Domain 2 includes ontologies from the Web directory domain [16], while Domain 3 contains four XML schemas from [18] belonging to two different domains. More details about data sets in Table I can be found in [10], [16]. Table I also shows that the number of our test data set is 73 data sources having elements ranging between 8 and 3306.

B. Evaluation strategy

In order to evaluate our approach, we carried out two sets of experiments. The first set is used to study the effect of each schema feature on the approach performance. To this end, we utilized only 13 data sources from Table I belonging to four different domains. The second set of experiments has been used to generally evaluate the approach.

The quality of clustering solutions has been verified using two common measures [2]: (1) *FScore* as a quality measure, and (2) *response time* as an efficiency measure. FScore is a trade-off between two popular information retrieval metrics, *precision P* and *recall R*. FScore combining precision and recall with equal weights for the given cluster C_i is defined by, $FScore_i = 2 * \frac{P_i * R_i}{P_i + R_i}$. The FScore of the overall clustering approach is defined as the sum of the individual class FScores weighted differently according to the number of XML data in the class

$$FScore = \frac{\sum_{n_i}^{k} n_i * FScore_i}{n} \quad (6)$$

where k, n_i and n are the number of clusters, the number of web data sources in a cluster C_i, and the number of web data source respectively. A good clustering solution has the FScore value closer to one.

C. Experimental Results

1) First experiment set.: In this set, we conducted three experiments. The first 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 experiment. 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 is then increased 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 are then merged to form a new level in the cluster hierarchy.

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. Precision, Recall, and FScore 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 II. 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 II also shows that both using the term vector and using both vectors produce a perfect cluster solution (FScore=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.

2) Lessons learned:

TABLE II: Performance result

	Quality			Run time	
	P	R	FScore.	$dist_{max}$	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

- The results from the first set, Table II, indicate that exploiting one feature of schema graph elements is not sufficient to get a good performance. This necessitates the need to utilize and combine both features
- Exploiting the term feature outperforms exploiting the structural feature from the quality point of view, while using the structural feature outperforms the term one from the performance point of view.
- There is a growing need to combine both linguistic and structural features in order to get a trade-off between the clustering quality and the clustering performance. Furthermore, it should be noted that the combination process should be biased towards the term feature vector (i.e. w in Eq. 1 should be greater than 0.5)

3) Second experiment set.: Motivated by lessons learned from the results of the first set of experiments, we conducted another set of experiments using the whole data set and capturing the term feature vector, the structural feature vectors, and both of them. In this set, we study the effect of combining both the term vector and the structure vector on the clustering solution. We report the clustering quality (FScore) and response time (number of iterations needed to build the cluster hierarchy) as a function of weighting values. Results are summarized in Figs 6& 6.

Fig. 5 shows the effect of the combining process on the clustering quality (FScore). The figure illustrates that with $w = 0$, i.e., only the structure vector is considered, the clustering quality has a low FScore value (0.5). Increasing the weighting value, i.e. considering the term vector with the structure vector, improves the clustering quality. The best clustering quality reaches 0.89 at $w = 0.8$ and it then reduces with increasing the weighting coefficient. The figure also demonstrates that when $w = 1$, i.e. considering only the term vector, the clustering quality has a FScore value of 0.83.

The efficiency of the clustering solution is represented in Fig. 7. It shows the number of iterations needed to build the cluster hierarchy w.r.t. the weighting coefficient (w). The figure shows that using only one feature vector ($w = 0$ or $w = 1$) need much time to construct the cluster hierarchy. Considering both features reduces the effort needed to build the hierarchy. The figure also illustrates that this effort reduces as the weighting coefficient increasing. The minimum number of iterations is 385 at $w = 0.9$, while it is 417 at $w = 0.8$.

VI. 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.

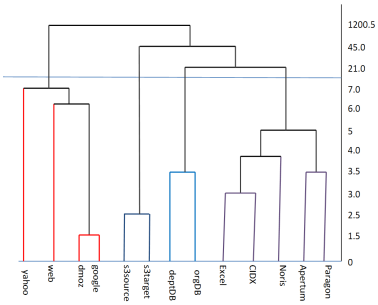

Fig. 3: Using name feature.

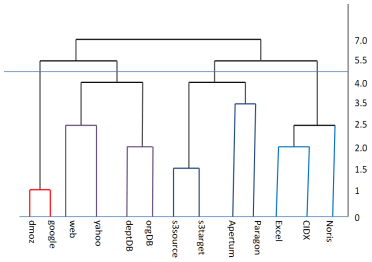

Fig. 4: Using structure feature.

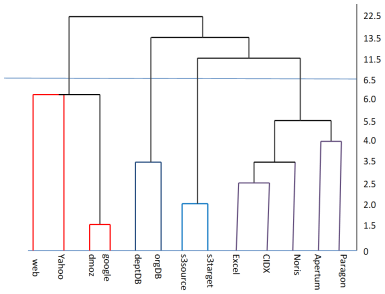

Fig. 5: Using both features.

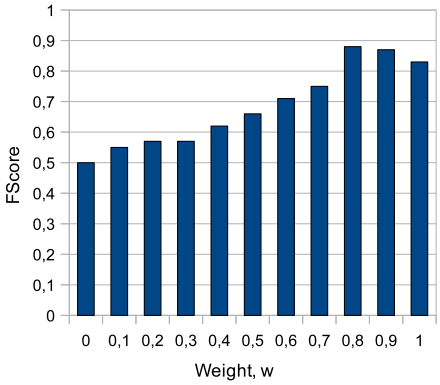

Fig. 6: FScore.

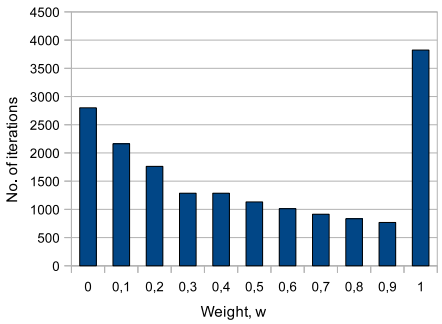

Fig. 7: No. of iteration.

The approach is generic, can deal with different data models and does not depend on human intervention. The approach combined both the linguistic and structural features of data sources to enhance schema similarity computation. We carried out an intensive 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.

REFERENCES

[1] C. C. Aggarwal, N. Ta, J. Wang, J. Feng, and M. J. Zaki. Xproj: a framework for projected structural clustering of XML documents. In *KDD 2007*, pages 46–55, 2007.

[2] A. Algergawy, M. Mesiti, R. Nayak, and G. Saake. XML data clustering: An overview. *ACM Computing Surveys*, 43(4), 2011.

[3] A. Algergawy, R. Nayak, and G. Saake. Element similarity measures in XML schema matching. *Information Sciences*, 180(24), 2010.

[4] A. Algergawy, E. Schallehn, and G. Saake. A schema matching-based approach to XML schema clustering. In *iiWAS2008*, 2008.

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

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

[7] M. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7, 2001.

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

[9] T. Dalamagasa, T. Cheng, K.-J. Winkel, and T. Sellis. A methodology for clustering xml documents by structure. *Information Systems*, 31:187–228, 2006.

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

[11] Y. Hafri, C. Djeraba, P. L. Stanchev, and B. Bachimont. A markovian approach for web user profiling and clustering. In *PAKDD*, 2003.

[12] M. L. Lee, L. H. Yang, W. Hsu, and X. Yang. Xclust: Clustering XML schemas for effective integration. In *CIKM'02*, pages 63–74, 2002.

[13] 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.

[14] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Y. Halevy. Google's deep web crawl. *PVLDB*, 1(2):1241–1252, 2008.

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

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

[17] R. Nayak and W. Iryadi. XML schema clustering with semantic and hierarchical similarity measures. *Knowledge-based Systems*, 2007.

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

[19] J. A. R.T. Marler. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 2004.

[20] T. Tran, R. Nayak, and P. Bruza:. Combining structure and content similarities for XML document clustering. In *AusDM 2008*, pages 219–226, 2008.

[21] A. Vakali, G. Pallis, and L. Angelis. *Personal Information Retrieval and Access*, chapter Clustering Web Information Sources. 2008.

[22] N. Yuruk, M. Mete, X. Xu, and T. A. J. Schweiger. AHSCAN: Agglomerative hierarchical structural clustering algorithm for networks. In *ASONAM09*.