Template Extraction from Heterogeneous Web Pages with Cosine Similarity

Kulkarni A. H.
M.B.E.Society's College of Engineering,
Ambajogai,
Maharashtra

Patil B. M.
M.B.E.Society's College of Engineering,
Ambajogai,
Maharashtra

ABSTRACT
Now a day's detection of templates from a large number of web pages has received a lot of attention. Template detection technique improves the performance of clustering, classification & search engines. In our work we proposed a novel algorithm by using cosine similarity based Template Extraction. We are using the cosine similarity approach to cluster the web documents. With the help of underlying structure of web documents we found the template for individual cluster. Our experimental evaluation show that our approach is effective in terms of computing Time and Clustering cost.

Keywords
Template Extraction, TEXT_MDL, TEXT_MAX, Cosine similarity clustering.

1. INTRODUCTION
The World Wide Web is one of the huge data repositories ever available in overall world. Many more Web data extraction techniques are developed in order to achieve the efficient access to relevant data from this repository. several techniques has been developed to solve the problem of Web data extraction but their use is still not spread, cause these requires more human interaction and doesn't provide the adequate results. Now a day's most of the information is stored in text databases. This information consist large collection of Heterogeneous web pages. Templates extraction is done from this collection of heterogeneous web pages. The problem of extracting template has been studied in [7], [9], [10]. In these all problems due to the assumption that all the web documents are generated from single template, proposed solutions are only applicable where collected web documents are guaranteed to belongs to a common template. To overcome this limitation extraction of template from the collection of web pages which belongs to different templates was also studied. In the solution of heterogeneous collection the web documents that belongs to same template are grouped together which will increase the dependency of extracted templates on quality of clustering. This paper presents algorithm for extracting templates from the collection of heterogeneous web pages. Clustering of documents depends on similarity of underlying structure of templates. To find out the underlying similarities between documents cosine similarity approach is included. Document clustering is depending on cosine similarity results. Generated clusters are themselves represents the extracted templates.

2. RELATED WORK
Template extraction problem is broadly classified into the Data mining. Initially the template extraction problem was studied by yossef & rajagopalan [4]. Both of them focused on the tags available in html documents to extract the template. But any content might be a part of template, since considered every word equally in solution. Vieira et al. [5] proposed an algorithm in which documents are represented in trees format and then found the underlying similarity. Reis et al. [6] suggested a tree-edit distance to cluster documents. Tree edit distance provides way to evaluate underlying structural similarities between web pages. Tree edit distance is nothing but the minimal set of operations required to transfer TA into TB.

2.1 RTDM Algorithm [6]
RTDM algorithm is used for determining a new type of mapping that calls Restricted Top-Down Mapping. It is based on a post-order traversal of the trees. The worst case of the RTDM algorithm occurs when the two trees being compared are all identical, except for their leaves. According to RTDM, the extraction task is done in four different steps:

- Web page clustering
- Retrieve extraction pattern
- Data matching
- Data labeling.

2.2 DOM Approach
This describes about the Document Object Model (DOM).DOM includes the simple way to represent HTML documents into the tree format [1]. The Document Object Model is a platform that will allow the application programs to access and modify the style, structure and content of documents at run time. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model. The DOM provides a framework to access XML or HTML web documents. The DOM also includes standard way to represent an HTML document in to tree structure. First node of the tree is always defines Document node and every HTML tag and text in the HTML element are treated as an element node and text nodes respectively. Every HTML attribute is an attribute node. Since all words are equally treated as defined, any type of node can be a part of a template. For instant representation of simple HTML document d in Fig. 1 into DOM tree is illustrated in Fig. 2.

```
< html >
< body >
    < hr >
    < b > first column < /b >
< /body >
< /html >
```

Fig 1: Sample Web Document *d*

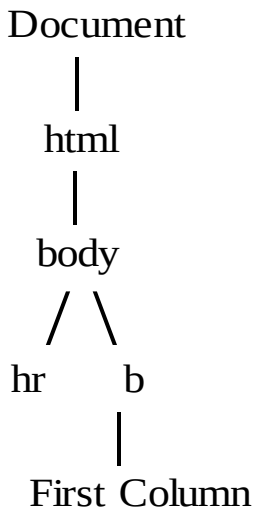

Fig 2: DOM tree of *d*

In a DOM tree, we can write a path of any node as follows: in Fig.2 the path of a node "first column" is "Document\<html>\<body> \< b>\ first column."

2.3 Essential Paths Generation

Given a web document collection $D = \{d1, d2, d3..dn\}$, where n represent any integer. Consider Pd as a set of all paths in di. Except the document node, cause a document node is the common node available in all DOM trees hence it should not be included in Pd set. To reduce the similarity costs do not considering all the paths of document di. Instead of that select some *essential paths* from the set Pd. *essential paths are* the path whose support value is greater than or equal to some threshold. The *support* value of a path Pi is equal to number of documents from set D including path Pi. Set individual threshold tdi for each document di. Set Minimum support value for the path pi where $pi \in pd$ and set this minimum support value as a Threshold tdi for document di. Notice that the threshold values of two different documents may be different. If any path $pi \in di$ and support value of path pi is greater than or equal to threshold value of document di then, path pi is called as essential path of document di.

2.4 Matrix Representation for Clustering

Clustering is the process of examining a collection of "points," and grouping the points into "clusters" according to some distance measure. In a text document clustering we use the similarity matrix representation. An approximation or representation to a matrix A can be thought of as a system which captures the most "important" information in A. Obtaining such representation usually involves a tradeoff among accuracy, the memory space occupied by the representation, and the time required to obtain the representation. These three aspects of the representation are usually in conflict, and must be tailored to fit the desired application.. Initially we have clustered set C and document set D. $C = \{c1, c2.....cn\}$ Where each $ci = \{Di, Ti\}$, ti is collection of essential paths which represents template of cluster ci and di is collection of documents which belonging to cluster ci. Notice that one document can enter into single cluster only. Relation between the documents and there essential paths can be represented by ME matrix. To represent clustering information MT and MD matrices are useful, where in MT metrics rows and column define essential paths and clusters respectively and rows and columns in MD denotes cluster and it's member documents. MΔ matrix represent the difference between(MT*MD) and ME.[8]

2.5 Minimum Description Length Principle:

To deal with unknown number of templates & to search best partitioning among all possible partitions, MDL principle is applied. The MDL principle recognized that the best model generated from data set is the one which minimize the cost of 1) Representation of mode in bits and 2) The number of bits required to encode the data with model. In matrix representation MT, MD & MΔ according to probabilities of 0, 1,-1 we can calculate the number of bits i.e. MDL cost of cluster set C.

2.6 Clustering with MDL cost:

This module describes the clustering of documents using MDL cost [1]. Document set D is the input parameter for this model where as cluster set C is the output. In cluster set C, ci is the i^{th} cluster represented by $ci = \{Di, Ti\}$ where Ti are the template paths and Di is the collection of member Documents. An MT and MD matrix provides the information about clustering mode C and goodness measurement is accomplished with MDL cost. TEXT-MDL is an agglomerative hierarchical clustering algorithm which starts it's execution by considering a single document as a individual cluster. Initially in this module the first best pair of documents whose MDL cost is minimum than other pairs is found with the help of GetBestPair procedure. In all iteration in GetBestPair the all possible pairs are taken and the MDL cost of that pair is compared with minimum MDL cost. If current MDL cost is minimum one then current pair is the best pair to cluster. Pairs are continuously merged till there is no further reduction is possible. To calculate the MDL cost of the pair $GetMdlCost(ci, cj, C)$ is called in GetBestPair method, where current pair to be merged is ci with cj and clustering model C is going to upgrade by first best pair. After the first best pair merging scale of the MDL cost reduction affects to remaining clusters in mode C, therefore GetBestPair should recalculate the MDL cost between the first best pair and all other clusters in C. The above two steps to find out the Best pair to merge and again the MDL cost of current best pair will affect to remaining clusters will increase the time complexity of algorithm. Hence it is not good practice to use TEXT-MDL algorithm with large number of web documents.

2.7 Clustering with MinHash:

In the Extended Minhash algorithm MDL cost calculation is depend on the *signature* of the documents. *Signature* of the documents is calculated by assigning the random ranks rj to individual elements of universal set U. Let set s $\in$ U & $\prod = \{\prod 1, \prod 2.....\prod L\}$ is the random permutation on set U. then the signature of set S is the collection of min ranks i.e. rj which is denoted by $\{rj \mid rj \in S\}$ in $\prod$ i permutation. Note that here $\prod$ should be the minimum it means maximum number of permutations should equal to number of intersection elements between any numbers of sets. Consider

two set s1 & s2 are the two documents and sig_{s1}, sig_{s2} are the signatures of documents s1, s2 respectively. With the help of signature we can found the number of essential paths i which are present in both the documents. The input parameter for this model is documents set D and output is cluster set C. Initially all documents are treated as a one cluster. Calculate the signature of each document. Merge the documents having the same signature. To find the best pair to merge we reduce the search space with the help of Jaccard's coefficient [2]. Given cluster ci to find out best pair with ci consider only those clusters whose Jaccard's coefficient is maximal with ci . Note in the previous model this search space is equal to all other clusters in set C. Now we can start to find the first best merging pair in GetInitBestPair method. To calculate the MDL cost we have to calculate the number of 1's in MT,-1 & +1 in MΔ according to Lemma3 [8]. And finally we can merge the two clusters whose MDL cost is the minimum one. We repeat the procedure of GetInitBestPair in GetBestPair. Complexities of GetInitBestPair and GetBestPair depend on the number of clusters selected in maximal Jaccard's coefficient

3. PROPOSED WORK

In this section of paper the native algorithm to cluster the heterogeneous documents is described. In the clustering process distance or similarity between given objects is calculated. With the help of cosine similarity approach the similarity between any two vectors is calculated. If similarity between any two documents is equal to 1 then one can say both document are similar to each other. Example 1 will illustrate the cosine similarity calculation.

Example 1:

Feature of Fruit	Sphere shape	Sweet	Sour	Crunchy
Object A=Apple	Yes(1)	Yes(1)	Yes(1)	Yes(1)
Object B= Orange	Yes(1)	Yes(1)	Yes(1)	No(0)

$$A = \{1,1,1,1\}$$
$$B = \{1,1,1,0\}$$
Dot Product :
$$A * B = w1 * w2 + x1 * x2 + y1 * y2 + z1 * z2$$
$$= 1*1 + 1*1 + 1*1 + 1*0$$
$$= 3$$
the norm of each vector (their length) is
$$| A | = (w1 * w1 + x1 * x1 + y1 * y1 + z1 * z1)\text{^}1/2$$
$$= (1+1+1+1)\text{^}1/2$$
$$= 2$$
$$| B | = (w2 * w2 + x2 * x2 + y2 * y2 + z * z2)\text{^}1/2$$
$$= (1+1+1+0)\text{^}1/2$$
$$= 1.732050888$$
$$| A | * | B | = 3.464101615$$
$$\text{Cosine Similarity} = A * B/(| A | * | B |)$$
$$= 3/3.464101615$$
$$= 0.866$$

The input to the algorithm is set of web documents D and output is the clustered templates C. one document can enter in one cluster only. To simplify the clustering calculation the web document is represented in DOM tree format [5]. After that *essential paths* of this web document is extracted and represented into the ME matrix [8]. Each column & row in ME represents the document and the essential paths of that document respectively. Cluster the documents whose essential path set is similar. Initially for each $di \in D$ one has to find out the cosine similarity with remaining all other documents. In order to calculate the cosine similarity, compare the presence or absence of essential paths between the documents. This information is present in ME matrix. To calculate similarity between document d1 & d2 take the first and second column of ME matrix and then find out the similarity as calculated in Example1. Similarly calculation process will repeat for all the clusters present in set C. As illustrated in example2. Number of similarity calculation between all the pairs in set D is equal to i*(i-1) where i=number of documents in set D.

Example 2:

Let $C = \{(c1 = d1), (c2 = d2), (c3 = d3), (c4 = d4)\}$ then similarity calculation for document *d1* should be in between $\{d1-d2, d1-d3, d1-d4\}$. For document d2 pairs like $\{d2-d1, d2-d3, d2-d4\}$. For document *d3* pairs should be $\{d3-d1, d3-d2, d3-d4\}$. Finally for d4 document pairs are like $\{d4-d1, d4-d2, d4-d3\}$.

According to constraint one document can enter into single cluster one has to select some best pair to cluster among i*(i-1) pairs. For that threshold value tdi is calculated. Note that there is no need of external input parameter to set threshold; in proposed algorithm threshold value is calculated as the average of cosine similarity values. Select only that pair whose cosine similarity is greater than or equal to tdi . Arrange all the selected pairs by maximum cosine similarity

value. Now the clustering process start with the set (say *similarity set*) of pairs such that $(ci, cj, \cos ine\_similarity\_value)$. The first pair in this is the best pair to merge in which documents di & dj is approximately similar to each other. Documents di , dj will form the new cluster ci ; Note here in the cluster ci only documents are included, to generate the template path one has to follow the template path generation step which is based on Theorem2 [8]. After merging the documents di & dj in cluster i remove all the entries of di & dj From the *similarity set.* Repeat the clustering and removing steps till there is a not a single entry in *similarity set*. At last cluster those documents which are near to each other. Few documents are such type which are not related to any other document hence they are not the part of *similarity set* such documents are consider as a single template & template path for that can be only the essential paths of same document.

```
1. C = {c1, c2, …, cn} with ci = (E(di), {di});
2. Merge all clusters with the same E(di).
3. For each ci in C do {
4. ctemp = C - ci;
5. For each cj in ctemp {
6.    (ci, cj, cosine_similarity) = calculate similarity value
      for the pair ci, cj;
7.    Similarity_Set = {ci, cj, cosine_value};
8.    count++;
9  }}
10 /* Set threshold value tdi
      tdi = ∑ cosine_ value / count
11. Descard all the entries from Similarity_Set whose
    cosine value is less than tdi.
12. Arrange the Similarity_Set in descending order.
13. while Similarity_Set is not Empty{
14 /* pick up the ci, cj one after othr from Similarity_Set
15. ci_temp = ci ; cj_temp = cj;
16. ck = ci ∪ cj;
17. Remove all the entries of ci & cj from Similarity_Set .
18 Similarity_Set = Similarity_Set -{ci, cj};
19 C = C -{ci, cj} ∪ ck;
20}
```

Fig 3: Cosine similarity algorithm.

Fig 3 shows an algorithm to cluster the web documents. Initially at line 1 each cluster ci includes the single document di and essential path set $E(di)$. Merge the cluster having the same essential path set. After that from line 3 to 9 calculate the cosine similarity between all the cluster pairs and insert these pairs into *similarity set*. To choose the best cluster pair to merge, set the threshold value tdi and the cluster pair whose cosine value is less then tdi are discarded from the *similarity Set*. At last from line 13 till 20 select the pairs of ci & cj one after other from *similarity set* & merges them. Discard all the entries of ci & cj from similarity set. Update the cluster set C.

4. EXPERIMENTAL RESULTS

Our proposed Cosine Similarity algorithm consumes less time to cluster the heterogeneous documents. We collected the web documents randomly from Internet. All experiments were performed on a Pentium(R) Dual-Core 2.oo GHz machine with 2 GB of main memory, running Windows XP operating system. All algorithms were implemented in JAVA with JRE version1.7.0.

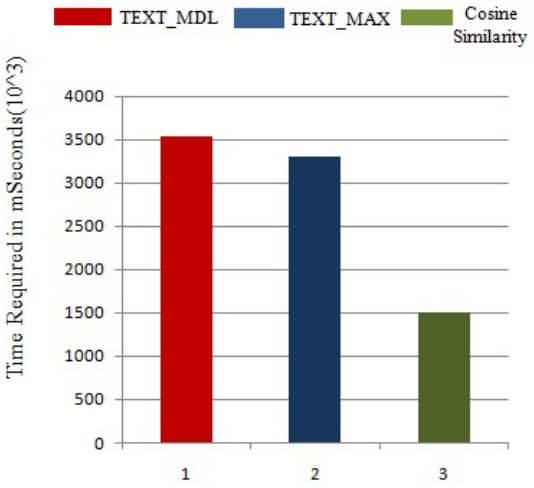

Fig 4: Bar Chart for time Analysis.

Fig 4 and 5 shows the graphical representation of the experimental results. To conform the effectiveness of cosine similarity algorithm the heterogeneous web pages are collected from sophisticated web sites like Amazon, yahoo. Google etc. same set of pages given as input to TEXT_MDL, TEXT_MAX & Cosine Similarity. Cosine similarity algorithm is more effective than remaining two algorithms in terms of Evaluation time and compactness of clustering. As the number of documents increased due to the large search space the clustering time is also get increased in TEXT_MDL, while in the TEXT_MAX though it is extended version, due to complex similarity calculations it consumes more time as compare to cosine similarity calculation.

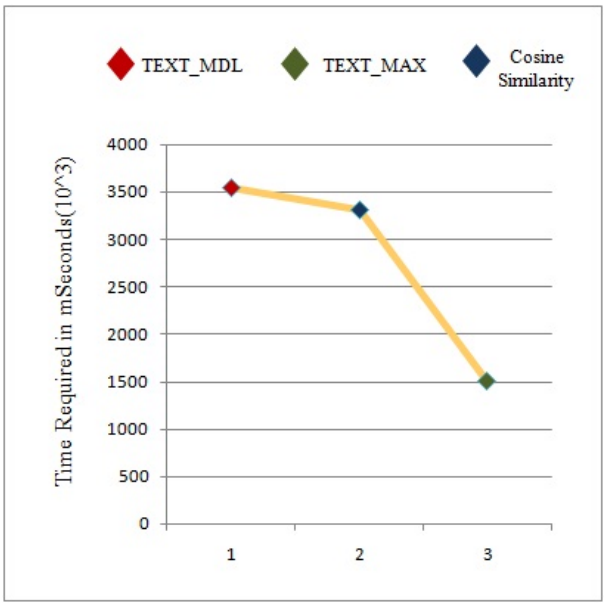

Fig 4: Line Chart for time Analysis.

5. CONCLUSION & FUTURE WORK

This system introduces an approach for template extraction from heterogeneous web documents. Proposed Cosine Similarity algorithm consumes less time to extract the templates as compare to existing algorithms like TEXT_MDL and TEXT_MAX. We used the cosine similarity distance formula to measure the similarity between web documents. Heterogeneous web documents are initially represented into a DOM structure. The underlying structural similarities between documents can easily find out with cosine distance formula. Closely related documents merged into clusters; simultaneously template is extracted from generated clusters. Experimental evaluation confirms effectiveness and robustness of algorithm with respect to time and compactness of clusters.

The two basic directions are encountered for future work. First one is to explore the simplicity, efficiency and effectiveness of cosine similarity approach. The other one is plan to analyze our work with some standard data set means the data sets which are used in EXALG & VINTS algorithm.

6. REFERENCES

[1] S. Zheng, D. Wu, R. Song, and J.-R. Wen, "Joint Optimization of Wrapper Generation and Template Detection," Proc. ACM

[2] SIGKDD, 2007.Z. Chen, F. Korn, N. Koudas, and S. Muithukrishnan, "Selectivity Estimation for Boolean Queries," Proc. ACM SIGMOD-SIGACTSIGART Symp. Principles of Database Systems (PODS), 2000.

[3] M. de Castro Reis, P.B. Golgher, A.S. da Silva, and A.H.F. Laender, "Automatic Web News Extraction Using Tree Edit Distance," Proc. 13th Int'l Conf. World Wide Web (WWW), 2004.

[4] Z. Bar-Yossef and S. Rajagopalan, "Template Detection via Data Mining and Its Applications," Proc. 11th Int'l Conf. World Wide Web (WWW), 2002.Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[5] K. Vieira, A.S. da Silva, N. Pinto, E.S. de Moura, J.M.B. Cavalcanti, and J. Freire, "A Fast and Robust Method for Web Page Template Detection and Removal," Proc. 15th ACM Int'l Conf. Information and Knowledge Management (CIKM), 2006.

[6] M. de Castro Reis, P.B. Golgher, A.S. da Silva, and A.H.F. Laender, "Automatic Web News Extraction Using Tree Edit Distance," Proc. 13th Int'l Conf. World Wide Web (WWW), 2004.

[7] A. Arasu and H. Garcia-Molina, "Extracting Structured Data from Web Pages," Proc. ACM SIGMOD, 2003.

[8] Chulyun Kim and Kyuseok Shim, Member, IEEE "TEXT: Automatic Template Extraction from Heterogeneous Web Pages"

[9] V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: Towards Automatic Data Extraction from Large Web Sites," Proc. 27th Int'l Conf. Very Large Data Bases (VLDB), 2001.

[10] K. Vieira, A.S. da Silva, N. Pinto, E.S. de Moura, J.M.B. Cavalcanti, and J. Freire, "A Fast and Robust Method for Web Page Template Detection and Removal," Proc. 15th ACM Int'l Conf. Information and Knowledge Management (CIKM), 2006.