Volume 3, No. 10, October 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

EXTRACTION OF STRUCTURED INFORMATION FROM UNSTRUCTURED OR SEMI- STRUCTURED MACHINE READABLE WEB PAGES

Vinod Kumar Raavi[*1] and Satya P Kumar Somayajula[2]
[*1] M Tech (Information technology) Student,
Avanthi Institute of Engineering & Technology, Narsipatnam, AndhraPradesh, India
[2]Asst.Professor, Dept of CSE,
Avanthi Institute of Engineering & Technology, Narsipatnam, Andhra Pradesh, India
chvinod48@gmail.com

Abstract: In now a days the extraction of structured information from unstructured or semi- structured machine readable documents extemporaneously plays a vital role hence many of the websites using ordinary templates with contents which produce the information to accomplish a well publishing productivity, but the major resource for extracting the information is WWW.Recently template detection approach has attained a lot of consolidation of effort in order to reform in various conditions like clustering and classification of web documents, performance of search engine as templates decrease the performance and the efficiency of web application for machines as a result of irrelevant template terms. We want to present a novel algorithm in this paper for extracting templates from a excessive number of web documents that are achieved from heterogeneous templates. By understanding the similarities of the basic template structure in the document we group the web documents so that template for each group has been simultaneously extracted. Hence the algorithms proposed in this paper can be considered as the best among all of the template detection algorithms.

Keywords: Template, Extraction, Information, DOM, Cluster.

INTRODUCTION

An HTML document can be naturally represented with a Document Object Model (DOM) tree as in Fig.1 , web documents are considered as trees and many existing similarity measures for trees have been investigated for clustering**[1]**. However, clustering is very expensive with tree-related distance measures. For instance, tree-edit distance has at least O(n1 Log n2) time complexity, Where n1 and n2 are the sizes of two DOM trees and the sizes of the trees are usually more than a thousand. Thus, clustering on sampled web documents is used to practically handle a large number of web documents**[3][9]**.

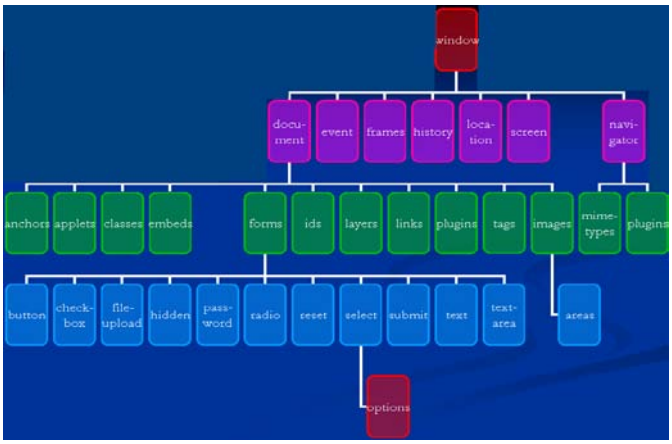

Figure.1 A simple Document Object Model (DOM)

The problem of extracting a template from the web documents conforming to a common template has been studied in**[4][10]**. Due to the assumption of all documents being generated from a single common template, solutions for this problem are applicable only when all documents are guaranteed to conform to a common template. However, in real applications, it is not trivial to classify massively crawled documents into homogeneous partitions in order to use these techniques **[5]**.The other area is the page-level template detection where the template is computed within a single document. Lerman et al. proposed systems to identify data records in a document and extract data items from them. Zhai and Liu proposed an algorithm to extract a template using not only structural information, but also visual layout information **[5]**. Chakrabarti et al. solved this problem by using an isotonic smoothing score assigned by a classifier [1]. Since the problem formulation of this area is far from ours, we do not discuss it in detail. Our algorithms to be presented later represent web documents as a matrix and find clusters with the matrix. Biclustering or coclustering is another clustering technique to deal with a matrix. Coclustering algorithms find simultaneous clustering of the rows and columns of a matrix and require the numbers of clusters of columns and rows as input parameters **[9][8].** However, we cluster only documents not paths, and moreover, the numbers of clusters of columns and rows are unknown.

In this paper we propose to represent a web document and a template as a set of paths in a DOM tree.As validated by the most popular XML query language XPATH, paths are sufficient to express tree structures and useful to be queried**[1]**. By considering only paths, the overhead to measure the similarity between documents becomes small without significant loss of information.

BACKGROUND WORK

There has been a lot of recent work related to Information Extraction. These can be classified along different dimensions: sources of information targeted (human vs.

machine generated), degree of automation, complexity of data extracted (flat vs. nested). Section 1 briefly mentioned some of the closely related work. We refer the reader to a recent survey and tutorial for more related work. Here we focus on highlighting the differences between our work and, ROADRUNNER and IEPAD. IEPAD uses repeating patterns of closely occurring HTML tags to identify and extract data**[1][2]**. The above technique is applicable to extracting data of a limited type: set of flat tuples, from each page. urther, since not all repeating patterns contain useful data, IEPAD uses various heuristic techniques to characterize those that do. Our work is most closely related to the ROADRUNNER. It uses a model of page creation using a template that is very similar to ours. It starts off with the entire first input page as its initial template. Then, for each subsequent page it checks if the page can be generated by the current template. If it cannot be, it modifies its current template so that the modified template can generate all the pages seen so far.

There are several limitations to the ROADRUNNER approach:

a. ROADRUNNER assumes that every HTML tag in the input pages is generated by the template. This assumption is crucial in ROADRUNNER to check if an input page can be generated by the current template. This assumption is clearly invalid for pages in many web-sites since HTML tags can also occur within data values. For example, a book review in Amazon could contain tags — the review could be in several paragraphs, in which case it contains _p_ tags, or some words in the review could be highlighted using _i_ tags. When the input pages contain such data values ROADRUNNER will either fail to discover any template, or produce a wrong template.
b. ROADRUNNER assumes that the "grammar" of the template used to generate the pages is union-free. This is equivalent to the assumption that there are no disjunctions in the input schema. The authors of ROADRUNNER themselves have pointed in that this assumption does not hold for many collections of pages. Moreover, as the experimental results in suggest, ROADRUNNER might fail to produce any output if there are disjunctions in the input schema.
c. When ROADRUNNER discovers that the current template does not generate an input page, it performs a complicated heuristic search involving "backtracking" for a new template. This search is exponential in the size of the schema of the pages. It is, therefore, not clear how ROADRUNNER would scale to web page collections with a large and complex schema.

This paper presented an algorithm, EXALG, for extracting structured data from a collection of web pages generated from a common template. EXALG first discovers the unknown template that generated the pages and uses the discovered template to extract the data from the input pages. EXALG uses two novel concepts, equivalence classes and differentiating roles, to discover the template. Our experiments on several collections of web pages, drawn from many well-known data rich sites, indicate that EXALG is extremely good in extracting the data from the web pages **[4]**. Another desirable feature of EXALG is that it does not completely fail to extract any data even when some of the assumptions made by EXALG are not met by the input collection. In other words the impact of the failed assumptions is limited to a few attributes. There are several interesting directions for future work.

The first direction is to develop techniques for crawling, indexing and providing querying support for the "structured" pages in the web. Clearly, a lot of information in these pages is lost when naive key word indexing, and searching is used. We indicate two specific problems in this direction. First, how do we automatically locate collections of pages that are structured? Second, is it feasible to generate some large "database" from these pages? Any technique for solving the latter problem has to be much less sophisticated than the one discussed here, possibly by sacrificing accuracy for efficiency**[2]**. Also when we work at the scale of the entire web we might be able to leverage the redundancy of the data on the web as in Brin. The second direction of work is to develop techniques for automatically annotating the extracted data, possibly using the words that appear in the template We presented a framework for classifier based page-level template detection that constructs the training data and learns the notion of "templateness" automatically using the site-level template detection approach **[4]**. We formulated the smoothing of classifier assigned templateness scores as a regularized isotonic regression problem on trees, and presented an efficient algorithm to solve it exactly; this may be of independent interest. Using human-labeled data we empirically validated our system's performance, and showed that template detection at the page-level, when used as a preprocessing step to web mining applications, such as duplicate detection and webpage classification, can boost accuracy significantly.

SYSTEM OVERVIEW

Modules Description:

Implemented Algorithms:

RTDM: We implemented RTDM since it is the\related work having the most similar problem formulation with us. It requires a training data set and the similarity threshold to decide the number of templates.

TEXT-MDL: It is the naive agglomerative clustering algorithm with the approximate entropy model It requires no input parameter.

TEXT-HASH: It is the agglomerative clustering algorithm with MinHash signatures discussed in it requires an input parameter which is the length of MinHash signature.

TEXT-MAX: It is the clustering algorithm with both MinHash signatures and Heuristic 1to reduce the search space. It requires the length of the signature as an input parameter.

A)

Real Life Data:

Data set 1 (D1): It is the data set used in EXALG. The documents are from nine templates and the number of

documents from each template is from 10 to 50. The total number of documents is 242.

Data set 2 (D2): It is the data set used in VINTS . The documents are from 100 templates and the number of documents from each template is 10. The total number of documents is 1,000.

Data set 3 (D3): We had crawled real life web documents for a week using Rank Mass Crawler Rank Mass theoretically guarantees that important part of the Web will be downloaded after crawling a certain number of pages and gives a high priority to important pages during the crawling process. The total number of documents is 100,000 (about 15 GB).

Performance Evaluation:

Clustering and Template Accuracy:

The ground truth of clustering of data sets D1 and D2 is known and we compare the clustering results of RDTM and our proposed algorithms with the ground truth. In order to quantify the accuracy of a cluster, we use the precision and recall values between a cluster and the closest ground truth cluster. The results are given in Table 2a, where # is the number of clusters found by each algorithm, P and R are the average precision and recall values of clusters

THE PROPOSED FRAMEWORK

Our template detection method is based on the repetition of text segments which are text nodes in DOM trees of web pages. We use a data structure called the text segment table to maintain the repetition information, i.e. the contents and DFs of text segments. We should note that when talking about the DFs of text segments, we must first clarify when two text segments are considered to be the same. In our framework, two text segments are same only when their contents are literally equal and they have the same DOM path. Whenever a new page is available, it will be passed through four steps:

Page Segmentation:

The segmentation process contains two steps:

a. A web page is divided into multiple blocks. Currently, we choose some html tags that usually determine the page layout as separators, these html tags are <TABLE>, <DIV>, etc.
b. Then each block is further divided into text segments by html tags, process instructions, and html comments.

Text Segment Table Expansion:

After page segmentation, text segments are used to updatethe text segment table. If a text segment already exists in the table, the DF of it will be increased by one. Otherwise, the text segment will be inserted into the table and its DF is initialized to one.

Template Detection:

Template detection occurs in block level. We search every text segment of a block in the text segment table, checking whether it is a template segment. We define template segments as text segments whose DFs are larger than or equal to 5. We can then calculate the template ratio of a block: template ratio = _lengths of template segments _lengths of all text segments If the template ratio of a block is larger than 0.7, we label the block as a template block.

Text Segment Table Shrinkage:

The text segment table will consume more and more storage if only the expansion step is applied. To control the storage use, we need to delete some text segments. The cost of deleting a text segment is defined as the times to classify a template segment as non-template segment be- cause of the deletion. For example, if a template segment appears after its deletion, it will be recognized as non-template segment. The cost of deleting a text segment is related to the DF and the future occurring times of the text segment. To minimize the cost of deletion, we allow text segments that have larger DFs to live longer than those with smaller DFs, because the former are more likely to occur in the future. We should note that we don't use the publish times or crawling times as the timestamps of pages and blocks, but we assign every page a page number and use it as the timestamp. The maximum living time of a text segment is then modeled by the logistic function: Tb is the maximum living time of a text segment which appears only once. df is the past DF of the text segment. Ten defines the upper bound of maximum living time of a text segment before a new occurrence comes, no matter what df is. When a text segment doesn't appear for t, it will be removed from the text segment table.

PROPOSED ALGORITHM

The TEXT-MDL algorithm:

We describe the implementation and performance of a compression-based model inference engine, MDL compress.

Procedure GetHashMDLCost(c_i, c_j, C)

Begin

a. $D_k := D_i \cup D_j$, $c_k := (\emptyset, D_k)$, $C' : C - \{c_i, c_j\} U \{c_k\}$;
b. *For each* π_q *in Π do {*
c. $r(sig\,D_k[q]) := min\,(r(sig\,D_i[q], r(sig\,D_j[q]))$;
d. *If* $r(sig\,D_i[q]) == r(sig\,D_j[q]$ *then*
e. $n(sig\,D_k[q]) := n(sig\,D_i[q] + n(sig\,D_j[q])$;
f. *Else* $n(sig\,D_k[q])$ *is from the less one;*
g. *}*
h. *Calculate* $\hat{\xi}(D_k, l)$ *by Equation(5);*
i. *Compute n(*D_k*,k) by Lemma 4;*
j. *Get* $P_r(1)$ *and* $P_r(-1)$ *in* M_T *and* M_Δ *by Lemma 3;*
k. *MDL := Approximate MDL cost of* C' *by Equation(1);*
l. *Return (MDL,* c_k *);*

end

The MDL-based compression produces a two part code of the training data, with the model portion of the code being used to compress and classify test data. We present pseudo-code of the algorithms for model generation and explore the conflicting requirements between minimizing grammar size and minimizing descriptive cost. We show results of a MDL model-based classification system for network traffic anomaly detection.

Agglomerative Clustering Algorithm:

The algorithm forms clusters in a bottom-up manner, as follows:

a. Initially, put each article in its own cluster.
b. Among all current clusters, pick the two clusters with the smallest distance.
c. Replace these two clusters with a new cluster, formed by merging the two original ones.
d. Repeat the above two steps until there is only one remaining cluster in the pool.

Thus, the agglomerative clustering algorithm will result in a binary cluster tree with single article clusters as its leaf nodes and a root node containing all the articles.

In the clustering algorithm, we use a distance measure based on log likelihood. For articles *A* and *B*, the distance is defined as

$$d(A,B) = LL(A) + LL(B) - LL(AUB)$$

The log likelihood *LL*(*X*) of an article or cluster *X* is given by a unigram model:

$$LL(x) = log \prod_{w \in X} px(w)^{ex(w)}$$
$$= \sum_{w \in X} e_{X(w)} log\ C_{x(w)} - N_x log N_x$$

Here, c X(w) and $PX(w)$ are the count and probability, respectively, of word *w* in cluster *X*, and N_X is the total number of words occurring in cluster *X*. Notice that this definition is equivalent to the weighted information loss after merging two articles:

$$d'(A,B) = (N_{A+}N_B)H(AUB) - (N_A H(A) + N_B H(B))$$

where

$$H(X) = -\sum_{w \in X} P_X(w) log\ P_X(w)$$

To avoid expensive log likelihood recomputation after each cluster merging step, we define the distance between two clusters with multiple articles as the maximum pairwise distance of the articles from the two clusters:

$$d(C_1,C_2) = \max_{A \in C_1, B \in C_2} d(A,B)$$

Where $C_1$and $C_2$are two clusters, and *A*, *B* are articles from $C_1$and C_2, respectively. Once a cluster tree is created, we must decide where to slice the tree to obtain disjoint partitions for building cluster-specific LMs. This is equivalent to choosing the total number of clusters. There is a tradeoff involved in this choice. Clusters close to the leaves can maintain more specifics of the word distributions. However, clusters close to the root of the tree yield LMs with more reliable estimates, because of the larger amount of data. We roughly optimized the number of clusters by evaluating the perplexity of the Hub4 development test set. We created sets of 1, 5, 10, 15, and 20 article clusters, by slicing the cluster tree at different points. A backoff trigram model was built for each cluster, and interpolated with a trigram model derived from all articles for smoothing, to compensate for the different amounts of training data per cluster. Then, the set of LMs that maximizes the log likelihood of the Hub4 development data was selected.

Given a cluster model set , $LM = \{LM_i\}$the test set log likelihood was obtained as an approximation to the mixture-of-clusters model:

$$P(w|LM) = \sum_i P(LM_i) * P(w|LM_i)$$
$$\approx P(LM_{i\leftarrow}) * P(w|LM_{i\leftarrow})$$
$$\propto P(w|LM_{i\leftarrow})$$

where $i^{\leftarrow} = \underset{i}{argmax}\ P(LM_i|A)$

and $P(LM_i)$ and $P(LM_i|A)$ are the prior and posterior cluster probabilities, respectively. In training, *A* is the reference transcript for one story from the Hub4 development data. During testing, *A* is the 1-best hypothesis for the story, as determined using the standard LM. Note that $P(w|LM)$ depends on the smoothing weights used to compute , $P(w|LM_i)$ which in turn determine which cluster a story is assigned to, which in turn determines the best smoothing weights.

Procedure *GetInitBestPair(c)*
begin
a. *Merge all clusters with the same signature ofMinHash;*
b. $MDL_{min} \coloneqq \infty$;
c. ***for each*** c_i ***in*** *C* ***do*** {
d. $N \coloneqq$ *clusters with the maximal Jaccard's coeff. with* c_i;
*/*If the maximal Jaccard's coefficient is 0,N is ∅*/*
e. ***for each*** c_j ***in*** *N* ***do***{
f. $(MDL_{tmp}, c_k) \coloneqq GetHashMDLCost(c_i, c_j, C)$;
g. ***if*** $MDL_{tmp} < MDL_{min}$ ***then*** {
h. $MDL_{min} \coloneqq MDL_{tmp}$;
i. $(c_i^B, c_j^B, c_k^B) \coloneqq (c_i, c_j, c_k)$;
a. }
j. }
k. }
l. ***return*** (c_i^B, c_j^B, c_k^B);
End

Procedure *GetHashBestPair(c_k, C)*
begin
a. $(c_i^B, c_j^B) \coloneqq$ *the current best pair;*
b. $c_k^B \coloneqq$ *a cluster made by merging* c_i^B *and* c_j^B;
c. $MDL_{min} \coloneqq$ *the current best approximate MDL cost;*
d. $N \coloneqq$ *clusters with the maximal Jaccard's coeff. with* c_k;
*/*If the maximal Jaccard's coefficient is 0,N is ∅*/*
e. ***for each*** c_l ***in*** *N* ***do***{
f. $(MDL_{tmp}, c_{tmp}) \coloneqq GetHashMDLCost(c_k, c_l, C)$;
g. ***if*** $MDL_{tmp} < MDL_{min}$ ***then*** {
h. $MDL_{min} \coloneqq MDL_{tmp}$;
i. $(c_i^B, c_j^B, c_k^B) \coloneqq (c_k, c_l, c_{tmp})$;
j. }
k. }

l. ***return*** $\left(c_i^B, c_j^B, c_k^B\right)$;

end;

Therefore, we jointly optimize smoothing and cluster assignment in an iterative procedure. First, the posterior probabilities of the smoothed cluster LMs given reference transcripts for a story were calculated. Then, stories with the highest posterior probability of a *same* cluster LM were merged. The interpolation weight for the cluster LM and the general LM was tuned by maximizing the likelihood of the segments in the story cluster corresponding to the cluster LM. These steps were iterated until all cluster assignments became stable and the interpolation weights converged.

SYSTEM ANALYSIS & DESIGN

The diagram shows a general view of how desktop and workstation computers are organized. Different systems have different details, but in general all computers consist of components (processor, memory, controllers, video) connected together with a *bus.*

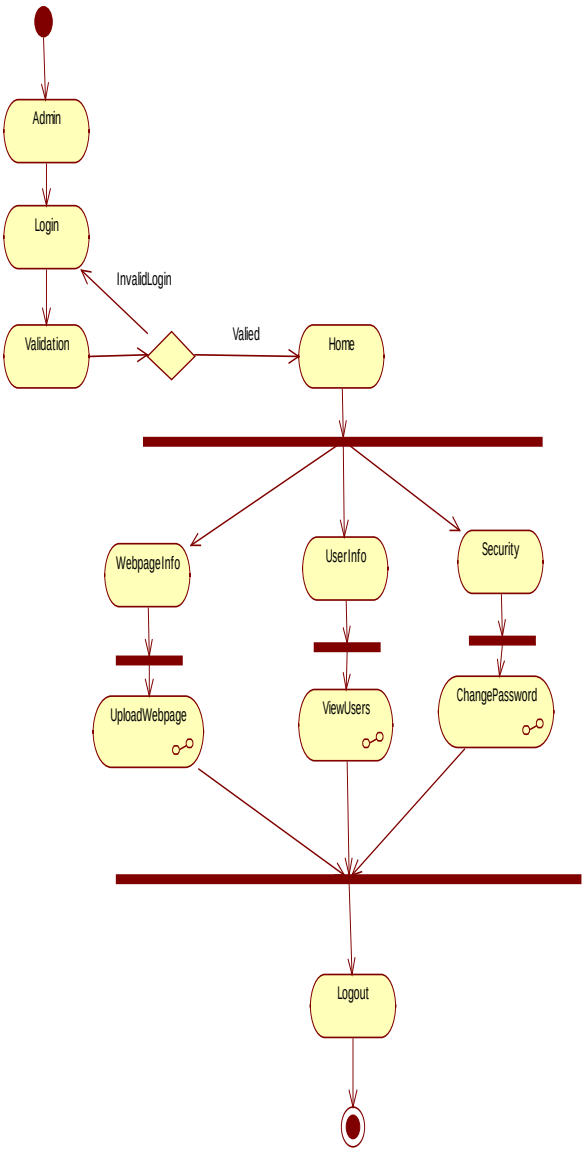

Figure. 2: The interoperaility Admin Activity diagram

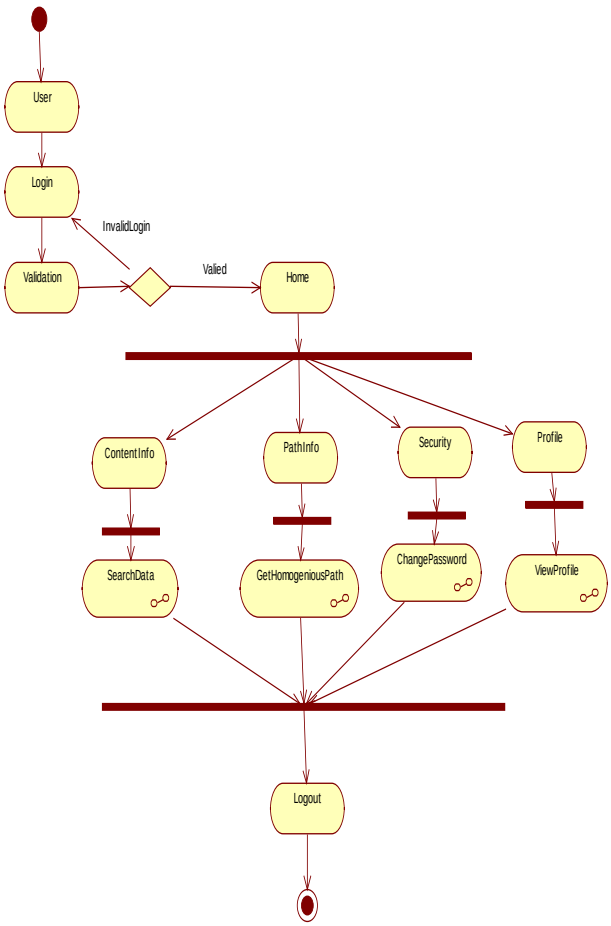

Figure .3: The interoperaility user Activity diagram

RESULTS

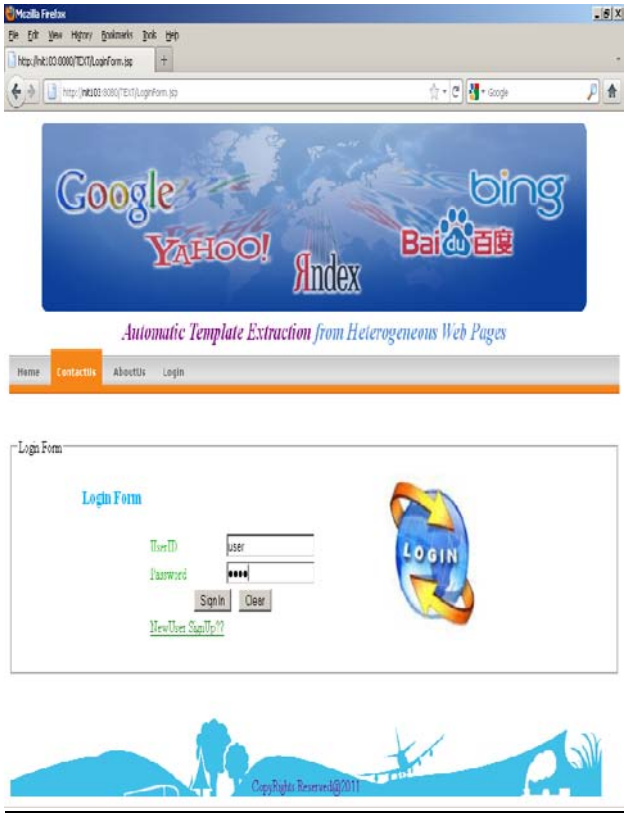

Figure .5: User Sign in Page

Figure.6 :User Search engine Page

Figure .8:Heterogenious path for different templates

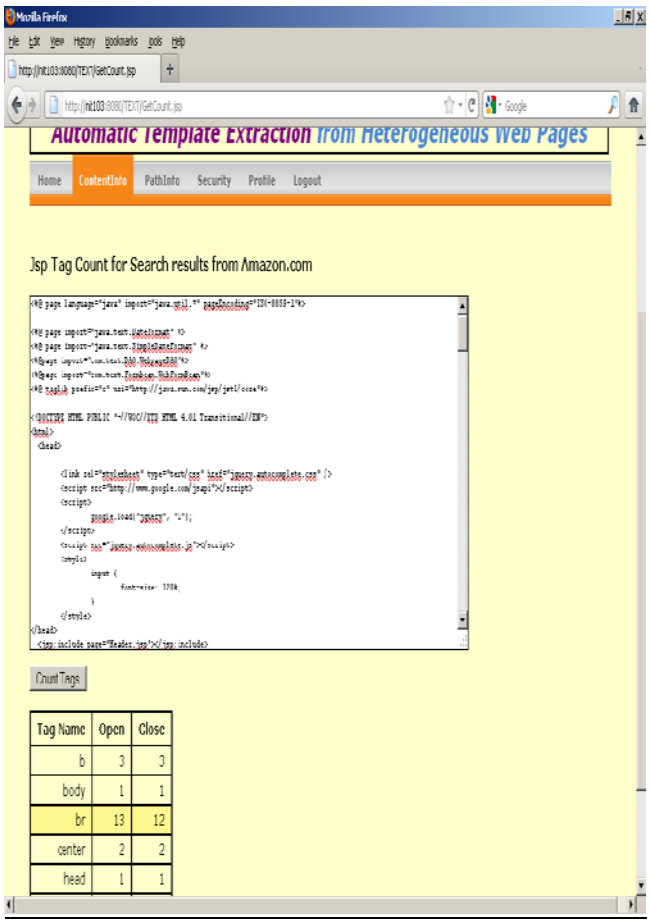

Figure.7: TagCountInformation of search result

Figure .9:Performance Reults for the cluster

CONCLUSION

In this paper we had introduced a novel approach of the template detection from different web documents. We applied the MDL principle for maintaining the anonymous number of clusters and also to select a good partitioning from all possible partitions of documents, later on we have extended MinHash approach just to fasten the clustering process. The effectiveness of our proposed algorithms can be confirmed by the experimental results with real life data sets.

REFERENCES

[1]. D. Chakrabarti, R. Kumar, and K. Punera, "Page-Level Template Detection via Isotonic Smoothing," Proc. 16th Int'l Conf. World Wide Web (WWW), 2007.

[2]. T.M. Cover and J.A. Thomas, Elements of Information Theory. Wiley Interscience, 1991.

[3]. B. Long, Z. Zhang, and P.S. Yu, "Co-Clustering by Block Value Decomposition," Proc. ACM SIGKDD, 2005.

[4]. 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.

[5]. H. Zhao, W. Meng, and C. Yu, "Automatic Extraction of Dynamic Record Sections from Search Engine Result Pages," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB), 2006.

[6]. Document Object Model (dom) Level 1 Specification Version 1.0, http://www.w3.org/TR/REC-DOM-Level-1, 2010.

[7]. 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.

[8]. H. Zhao, W. Meng, and C. Yu, "Automatic Extraction of Dynamic Record Sections from Search Engine Result Pages," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB), 2006.

[9]. M.D. Plumbley, "Clustering of Sparse Binary Data Using a Minimum Description Length Approach," http://www.elec. qmul.ac.uk/staffinfo/markp/, 2002.

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

Short Bio Data for the Author

Mr. VINOD KUMAR.RAAVI received the B.Tech degree from Vignan's Engineering College, Vadlamudi in 2009 and He is currently pursuing M.Tech in the Department Of Information Technology, Avanthi Institute of Engineering and Technology, Vishakhapatnam, JNTUK university. His research interests include Knowledge and Data Engineering, Web Technologies.

Mr. Satya P Kumar Somayajula is working as an Asst.Professor, in CSE Department, Avanthi Institue of Engg & Tech, Tamaram, Visakhapatnam,A.P., India. He has received his M.Sc(Physics) from Andhra University, Visakhapatnam and M.Tech (CST) from Gandhi Institute of Technology And Management University (GITAM University), Visakhapatnam, A.P., INDIA. He published many National and International Journals. His research interests include Image Processing, Networks security, Web security, Information security, Data Mining and Software Engineering.