Journal of Theoretical and Applied Information Technology
28th February 2014. Vol. 60 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SIMPLIFIED SCHEME FOR PERFORMANCE AUGMENTATION OF WEB DATA EXTRACTION

[1]G.NAVEENSUNDAR, [2]D.NARMADHA, [3]DR.A.P.HARAN
[1] *Karunya University, Department of CSE, Coimbatore,* [2]*Karunya University, Department of IT, Coimbatore,*
[2]*Park Engineering College, Aeronautical Engineering Department, Coimbatore*
Email: [1]naveensundar@karunya.edu , [2]narmadha@karunya.edu , [3]haran_pct@gmail.com

ABSTRACT:
Web mining is the application of data mining techniques to automatically discover and extract information from Web data. Furthermore, it uses the data mining techniques to make the web more profitable and to enhance the effectiveness of our interaction with the web. Users always expect maximum accurate results from search engines. But, unfortunately most of the web pages contain more unnecessary information than actual contents. The unnecessary information present in web pages is termed as templates. Template leads to poor performance of search engines due to the retrieval of non-contents for users. Therefore the performance of search engines can be improved by making web pages free of templates. Our method focuses on detecting and extracting templates from web pages that are heterogeneous in nature by means of an algorithm. Locality sensitive hashing algorithm finds the similarity between the input web documents and provides good performance compared to Minimum Description Length(MDL) principle and hash cluster process in terms of execution time.
Keywords:- *Cluster, Non-Content Path, Template Detection, Locality Sensitive Hash, Minimum Description Length*

1. INTRODUCTION

In recent years the development of World Wide Web exceeded all expectations. Nowadays there are several billions of pictures, HTML documents, and other multimedia files available via internet and the amount is still rising. But considering the striking variety of the web, retrieving interesting content has become a very daunting task. Web pages in the Websites are constructed in such a way that almost 50% of the data contains templates. This percentage is still increasing as time goes. Templates are a foundation on which actual content is built. From a user point of view, presence of templates is very much useful as they provide uniformity in look and feel of web pages. At the same time, presences of templates in very large amount in web pages compromise the performance of search engines. Also users are distracted from actual contents and are forced to access unimportant information from web sites. Hence it is required that the templates should be removed from web pages so that search engines can give good performance in terms of providing the most relevant information in response to user queries.

Many of the existing systems were based on the assumption that all the web pages under consideration are built using the same type of template. Such an assumption is not valid in most cases as web pages are built using different types of template structures. Hence this paper is based on the assumption that web pages under consideration are of different types. The structure of templates is different in those pages. A concept called clustering is proposed in this paper, in which documents belonging to same template structure are grouped in one cluster. A new algorithm is proposed for the purpose of clustering. A type of hashing may be performed prior to clustering so that performance in terms of execution time can be improved.

2. RELATED WORK

As per the method proposed by B. Adelberg, NoDoSE[1] Northwestern Document Structure Extractor (NoDose) is an interactive tool for semi automatically determining the structure of such documents and then extracts their data. The approach is called semi-automatic because it cooperates with the user to extract the data. The input to the extractor is text file or documents of same type. Using the GUI the user hierarchically decomposes the file based on

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DOM structure, outlining its interesting region and describing their semantics. The performance of system is fine for small files but it is not able to deal with large files and extract templates. Furthermore, there are different methods available for template detection and extraction. Many of the previous methods [2],[3],[4] were based on the assumption that all the web pages

ID	Path	Pathno
P1	Document\<html>	2
P2	Document\<html>\<body>	2
P3	Document\<html>\<body>\<h1>	2
P4	Document\<html>\<body>\ 	2
P5	Document\<html>\<body>\Gate	1
P6	Document\<html>\<body>\<h1>\IT	1
P7	Document\<html>\<body>\<h1>Park	1

belong to a common template structure. The use of factors like Tree-edit distance [3],[5] is very much expensive. A typical web page contains a title banner, list of links in right or left or both for site navigation and advertisements, a footer containing copyright statements, disclaimers or navigational links[6]. Mostly, meaningful content lies at the centre of the page. The design of web page is not standard for all web pages, consequently, a more robust and flexible content extraction tool is essential. Recent web pages have a cleaner architecture. They provide separation among visual presentation, real content and the interaction layers having abandoned the use of old structural tags and adopted an architecture that makes use of the style sheets and div or span tags [6]. This reduces the effectiveness of the old content extraction techniques.

Many existing approaches [2],[7],[8] use frequency of words as similarity measure. Template detection may be based on a threshold value [5] for the frequency of text in documents. Some of the previous approaches require large human intervention for collecting training examples[9] in order to distinguish between actual content and templates. A page-level[10] type of template detection detects templates on a page by page basis. The latest approach uses both frequency as well as a principle called MDL (Minimum Description Length) as decisive factors for detecting templates. MDLval is calculated which indicates the lowest number of bits required to represent a cluster. The cluster with the least MDLval is selected as the best cluster. The approach proposes an algorithm called Extract Template for clustering.

3. REPRESENTATION OF WEB PAGES

Web pages are usually represented as HTML documents. HTML documents can be represented in the form of DOM trees. Clustering requires some similarity measures for grouping. Existing systems use Tree-edit distance as a similarity measure but it is expensive because of its time complexity which is very much high. Hence the current system represents documents and templates with the help of paths of a DOM tree. This reduces the difficulty of finding the similarity of documents under consideration. The algorithms proposed in this paper represent web documents in the form of matrices.

Table 1: Paths and Pathno values

As an example consider two web documents represented using HTML tags shown in Table 2 and their corresponding paths and pathno values are shown in Table 1. Pathno of a path represents the number of documents in which the path occurs.

Table 2: Sample Web Document

<html> <body> IT </body>	</html> <html> <body> Park Gate </body>

4. IDENTIFYING NON-CONTENT PATHS

The web documents are given a threshold value known as least pathno threshold value. It is calculated as the mode of pathno values of paths in each document. A path is said to be a non-content path of a document, if the path is present in that document and it has the least pathno threshold value specified for the document. The documents are given a threshold value known as minimum support threshold value. It is calculated as the mode of support values of paths in each document. A path is said to be an essential path of a document d_i, if the path is contained in that document and it has the minimum support threshold The non-content path set of a document doc_i is represented as $NC(doc_i)$. A |PDOC| *|DOC| matrix MtNC with values 0/1 are used to represent web documents

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

where PDOC is the path set and DOC is the document set. A value of 1 at the i^{th} row and j^{th} column indicates that the path, $path_i$ is a non-content path of document doc_j. A value of 0 indicates that the path is a content path.

5. CLUSTERING USING MDL

To find the best cluster, a principle is used in this paper termed as Minimum Description Length (MDL) principle. According to MDL principle the cluster with lowest number of bits used to represent it is identified as the best cluster. It is termed as MDLval of the cluster.

5.1 MDLVAL Calculation

For a cluster model CL, the MDL value indicated as MDLval is represented as MDv(CL). It is calculated as the sum of MDL values of MtTEMP and MtDOC . The MDL values of MtTEMP and MtDOC are calculated as:

$$H(X) = \Sigma -P(x)\log_2 P(x) \quad (1)$$
$$x\epsilon \{0, 1, -1\}$$
$$MD_v(M_t) = |M_t|.H(X) \quad (2)$$

where H(X) is the entropy of a random variable X in the matrix, P(x) is the probability of 1's , -1's and 0's in the matrix. MDLval of a clustering model CL is calculated as:

$$MDv(CL) = MDv(Mt_{TEMP}) + MDv(Mt_{DOC}) \quad (3)$$

where $MDv(Mt_{TEMP})$ is the MDLval of matrix MtTEMP and $MDv(Mt_{DOC})$ is the MDLval of matrix Mt_{DOC}. MDL principle states that if 2 clustering models CL1 and CL2 are considered, the cluster with the lowest MDL value is taken as the best cluster. CL1 is taken as the best cluster when compared to CL2 if and only if MDv(CL1) is less than MDv(CL2).

5.1.1Clustering using TEXT-MDL Algorithm

TEXT-MDL algorithm takes a set of documents as input and produces a set of clusters as output. The decisive factor used for clustering is MDLcost. The TEXT-MDL algorithm is shown below.

Algorithm Extract Template (DOC)
begin
1. $CL:=\{cl_1, cl_2, \ldots cl_n\}$ with $cl_i=(NC(doc_i),\{doc_i\})$;
2. $(cl_i,cl_j,cl_k):=$FindBestCluster(CL);
3. While(cl_i,cl_j,cl_k) is not null do {
4. $CL:=CL-\{cl_i,cl_j\}U\{cl_k\}$;
5. $(cl_i,cl_j,cl_k):=$FindBestCluster(CL);}
6. return CL
End

procedure CalcMDLval(cl_i,cl_j,CL)
begin
1. $DOC_k:=DOC_i \cup DOC_j$;
2.$TEMP_k:=\{pathx|ndoc(path_x,DOC_k)>=|DOC_k|+ 1/2, pathxNC_k\}$;
3. $cl_k:=(TEMP_k,DOC_k)$;
4. CL':=$CL-\{cl_i,cl_j\}U\{cl_k\}$;
5. MDL:= MDL value of CL' ;
6. return(MDL,cl_k);
End

where CL indicates the whole clustering model, $cl_1,cl_2,\ldots$indicates individual clusters, $NC(doc_i)$ represents non-content path of doc_i,clk indicates newly formed clusters, $MDLval_{lowest}$ represents the lowest MDL value, TEMPk represents template paths and ndoc ($path_x$, DOC_k) indicates number of documents in which pathx is a non-content path. The algorithm is an agglomerative hierarchical clustering algorithm in which clusters are formed by grouping documents with similar structures. A set of documents are given as input to the algorithm. Initially each document is considered as separate clusters. When two clusters are clustered, there will be a change in MDLval. If MDLval is reduced as a result of merging two clusters, that cluster can be chosen as a best cluster. Such a best cluster is found out by the procedure FindBestCluster. MDLval is calculated by a procedure CalcMDLval (cl_i, cl_j, CL) where cli and cl_j are the clusters to be merged and CL is the current clustering present.ndoc ($path_x$, DOC_k) indicates the number of documents in which path, pathx is a non-content path.

5.1.2 Clustering using minhash

A cluster is chosen as the best one if it's reduction of MDL cost is maximum. MinHash is used to find the Jaccard's coefficient. If the coefficient is greater for some clusters then the MDLcost reduction will also be greater. The method helps in reducing the search space to a great extent when compared to TEXT-MDL approach. The procedures to find the best cluster using MinHash is given below.

procedure: GetInitBestPair(C)
begin
1. Merge all clusters with the same signature of MinHash;

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. MDL_{min}:=∞;
3. For each c_i in C do {
4. N :=clusters with the maximal Jaccard's coefficient with c_i;
5. for each cj in N do {
6. (MDL_{tmp},c_k):= GetHashMDLcost(c_i,c_j,C);
7. If MDL_{tmp}<MDL_{min} then {
8. MDL_{min}:= MDL_{tmp};
9. (c_i^B,c_j^B,c_k^B):=(c_i,c_j,c_k);
10. }}
11. return(c_i^B,c_j^B,c_k^B);

end

procedure: GetHashBestPair(c_k,C)

begin

1. (c_i^B,c_j^B):=the current best pair;
2. c_k^B :=a cluster made by merging c_i^B and c_j^B ;
3. MDL_{min}:= the current best MDLcost;
4. N :=clusters with the maximal Jaccard's coefficient with c_k;
5. for each c_l in N do {
6. (MDL_{tmp},c_{tmp}):= GetHashMDLcost(c_k,c_l,C);
7. If MDL_{tmp}<MDL_{min} then {
8. MDL_{min}:= MDL_{tmp};
9. (c_i^B,c_j^B,c_k^B):=(c_k,c_l,c_{tmp});
10. }}
11. return(c_i^B,c_j^B,c_k^B);

end

procedure: GetHashMDLcost(c_i,c_j,C)

begin

1. D_k :=D_i U D_j, c_k:=(Ø,D_k), C'={c_i,c_j} U {c_k};
2. for each Π_q in Π do {
3. r (sig_{Dk}[q] :=min(r(sig_{Di}[q]),r(sig_{Dj}[q]));
4. if (r(sig_{Di}[q])== r(sig_{Dj}[q]) then
5. n (sig_{Dk}[q] :=n (sig_{Di}[q]) + n (sig_{Dj}[q]);
6. else n (sig_{Dk}[q]) is from the less one}.
7. Calculate ε (D_k,l) ;
8. MDL :=MDLcost;
9. return(MDL,c_k);

end

In the MinHash algorithm, MDLcost is calculated using the procedure GetHashMDLcost. The signature values of input documents are considered and the minimum value is taken. The probability that a particular path is present in certain number of documents is then found out and based on that, the MDLcost is calculated. MDL_{tmp} the temporary MDLcost.

5.1.3 LSH algorithm

The enhanced algorithm when the hashing concept is included is given below. The current algorithm is the same as the basic approach with slight modifications.

Algorithm: TEXT-MDL(D)

begin

1. C:= {c_1,c_2,….c_n} with c_i=(E(d_i),{d_i});
2. (c_i,c_j,c_k):=GetBestPair(C);
3. While(c_i,c_j,c_k) is not empty do {
4. C:=C-{c_i,c_j}U{c_k};
5. (c_i,c_j,c_k):=GetBestPair(C);}
6. return C

end

procedure: GetBestPair(C)

begin

1. $MDLcost_{min}$:=∞;
2. For each pair(c_i,c_j) of clusters in C do {
3. (MDLcost,c_k):= GetLSHMDLcost(c_i,c_j,C);
4. If MDLcost<$MDLcost_{min}$ then {
5. $MDLcost_{min}$:= MDLcost;
6. (c_i^B,c_j^B,c_k^B):=(c_i,c_j,c_k);
7. }}
8. return(c_i^B,c_j^B,c_k^B);

end

procedure: GetLSHMDLcost(c_i,c_j,C)

begin

1. D_k:=D_i U D_j, C'=C- {c_i,c_j} U c_k;
2. Compute hash function.
3. Compare hash values of documents.
4. For any two points p and q that are close to each other, there is a high probability P_1 that they fall into the same bucket.
5. For any two points p and q that are far apart, there is a low probability P2<P_1 that they fall into the same bucket.
6. Compute Pr(1) and Pr(-1) in M_T and M_D .
7. MDL:= MDLcost .
8. return (MDL,c_k);

end

procedure: GetBestPair(c_k ,C)

begin

1. (c_i^B,$c_j^{B)}$:= the current best pair;
2. c_k^B:= a cluster made by merging c_i^B and c_j^B;
3. MDL_{min}:=the current best MDLcost;
4. For each c in C do {
5. (MDL_{tmp},c_{tmp}):= GetLSHMDLcost(c_k,c_l,C);
6. If MDL_{tmp}<MDL_{min} then {
7. MDL_{min}:= MDL_{tmp};

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

8. $(c_i^B, c_j^B, c_k^B) := (c_k, c_l, c_{tmp})$;
9. }}
10. return(c_i^B, c_j^B, c_k^B);

end

Here MDLcost is calculated by a procedure GetLSHMDLcost where, c_i and c_j are the clusters to be merged and C is the current clustering present. In GetLSHMDLcost, hash values are computed corresponding to the documents using hash function. The computed hash values are then compared for the documents and thus similarity is found out.

6. SIMPLIFIED DESIGN OF PROPOSED APPROACH

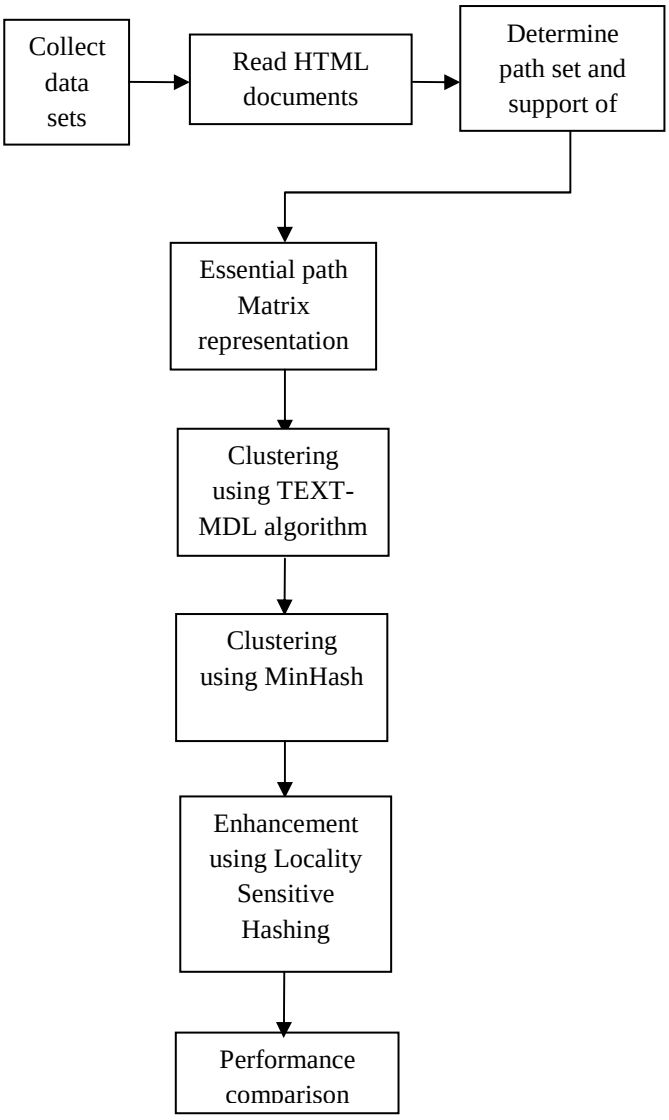

Figure 1: Simplified Scheme of proposed Approach

The simplified scheme of the complete process is shown in Figure 1. The input documents are read and path sets are determined. A block diagram of the whole process happening in the work is shown in Figure 1. Data sets are taken from five different web sites to ensure heterogeneity of the templates. The web pages are read and parsed using HTML parser. As a result of parsing, the paths are extracted and the support of the paths is determined. The Essential paths are the found out and represented in the form of a matrix. The process of clustering is then performed using TEXT-MDL algorithm.As a result of clustering, member documents and template paths in the clusters are determined. As a fast approximation of the above method, clustering is then performed using the MinHash concept and the corresponding clusters are determined. To improve the performance, clustering is then performed using the Locality Sensitive Hashing method and finally the performances of all the methods are compared.

7. EXPERIMENTAL ANALYSIS

The method proposed in this paper was implemented by Java JDK and Microsoft SQL Server 2000, and a personal computer with Windows XP was used for the evaluation experiments.

7.1 Data Sources and Results

Performance of the implemented methods is considered for analysis. The parameters used for performance analysis include execution time taken in seconds and the usage of memory. When compared with previous methods like RTDM, TEXT-MDL requires less execution time. When the fast approximation of TEXT-MDL which is TEXT-HASH is considered, it requires still less execution time. The memory needed for storage is also less for TEXT-HASH when compared with TEXT-MDL. In the case of Locality Sensitive Hashing approach, the execution time taken as well as the memory usage is observed to be again less than TEXT-HASH. Hence a very good improvement in performance is achieved.

To perform analysis, a set of five documents are taken and their results are analysed as shown in Table 3. As shown in the table, execution time taken is five seconds for TEXT-MDL, three seconds for TEXT-HASH and one second for Locality Sensitive Hashing for an input set of five documents. The memory required is 5232

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

bytes for TEXT-MDL and 3223 bytes for TEXT-HASH and 1242 bytes for Locality Sensitive Hashing for the same set of input documents.

7.2 Comparison between TEXT-MDL and TEXT-HASH

The results are analysed for different number of inputs including ten. In the case of TEXT-MDL, clusters are formed only on the basis of MDL principle. There is no measure of similarity in TEXT-MDL. While in the case of TEXT-HASH, hash values are used as a measure of similarity for finding the similarity of documents and then the clustering is performed based on the MDL principle. The time comparison graph is shown in Figure 2. As shown in Figure 2, the TEXT-MDL approach takes more execution time than TEXT-HASH for each and every input. It is mainly due to the fact that, the search space is reduced in the case of TEXT-HASH as the similarity between input documents are found out based on the hash values

Table 3: Performance Analysis

Method	No of documents	Execution time in seconds	Memory
TEXT-MDL	5	5	5232
	10	8	8324
	3	6	6452
	4	5	5124
TEXT-HASH	5	3	3223
	10	6	6452
	3	2	2334
	4	3	3452
Locality Sensitive Hashing	5	1	1242
	10	3	3124
	3	1	1234
	4	2	2124

The comparison between memory usage of TEXT-MDL and TEXT-HASH is shown in Fig 3.

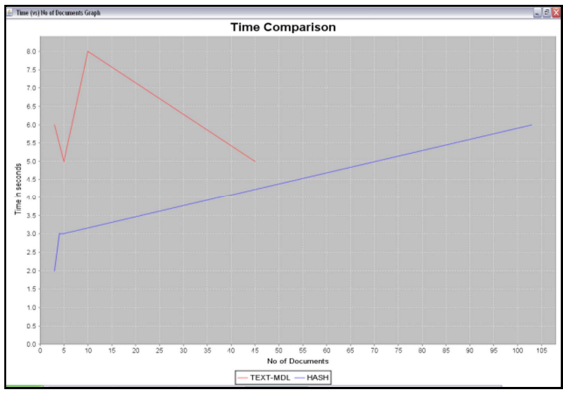

Figure 2 Time Comparison between MDL and Hash

As in the case of execution time, the memory usage is also less for TEXT-HASH than TEXT-MDL. The calculation of similarity between input documents helps in the storage of only less number of documents for clustering. The documents that are similar in hash values are stored and therefore, the number of documents which are getting stored for clustering will be less when compared to the number of documents that are getting stored in the case of TEXT-MDL, which considers each and every document for clustering. As a result, the total memory used by TEXT-HASH is very much less when compared to TEXT-MDL, which does not use any similarity measure to calculate the similarity between documents. Hence the performance of TEXT-HASH is very much better when compared to TEXT-MDL.

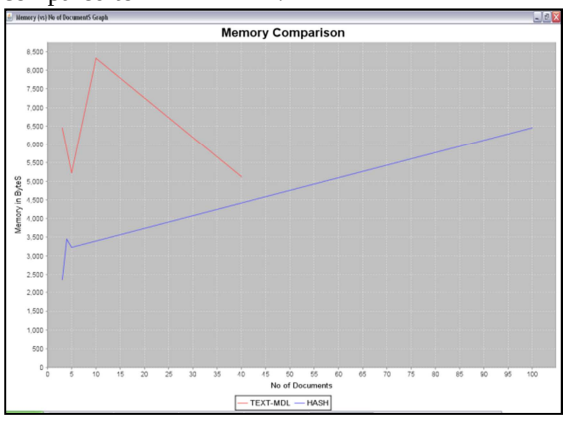

Figure 3 Memory Comparison between MDL and Hash

7.3 Comparison between Text-HASH and LSH

In the case of TEXT-HASH, similarity between the documents is found out first based

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

on the similarity in hash values of documents. As a result, only similar documents are considered for clustering as opposed to TEXT-MDL in which, each and every documents are clustered and tested based on least MDLcost. Hence the search space is reduced for TEXT-HASH. Therefore a drastic reduction in execution time and memory is observed in TEXT-HASH.

In the case of Locality Sensitive Hashing also, similarity between documents are found out first based on similarity between hash values of documents. As a result, the similar documents are considered for clustering as in the case of TEXT-HASH. Hence the search space is reduced in this type of approach.

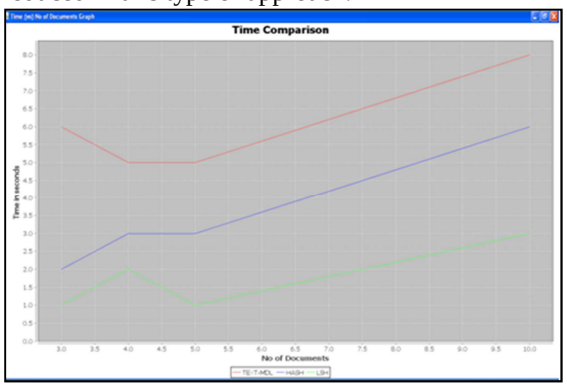

Figure 4 Time Comparison between MDL, HASH and LSH

The memory comparison graph is shown in Figure 5.A drastic reduction in memory usage is observed in Locality Sensitive Hashing when compared to TEXT-MDL and TEXT-HASH. The performance of TEXT-HASH lies in between TEXT-MDL and Locality Sensitive Hashing.

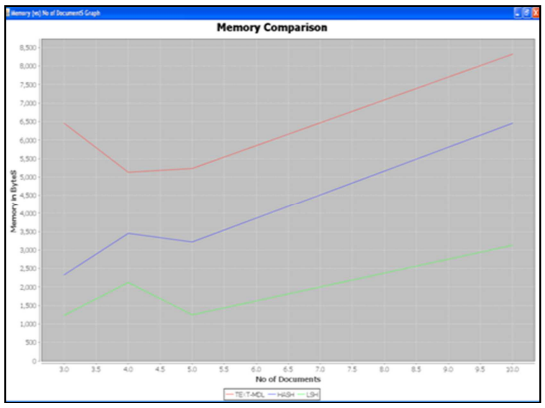

Figure 5 Memory Comparison between MDL,HASH,LSH

8. CONCLUSION AND FUTURE WORK

The current work involves extracting templates from web pages automatically. The implementation of the work is divided into six modules and it is successfully completed. As a continuation of the implementation process, an approach termed as Locality Sensitive Hashing is also proposed and it is implemented successfully with a very high improvement in performance. The proposed approach can detect and extract templates from heterogeneous web pages. The algorithms proposed in this work will be able to extract templates and make web pages free of such irrelevant information. Search engines will be able to retrieve the best pages for the users based on their queries.

The web pages taken as input are taken from different web sites and they are static in the current work. Therefore as a future work, the web pages can be made dynamic and the clustering process can then be performed on those pages thereby extracting template paths from them.

REFERENCES:

[1] B. Adelberg,"NoDoSE – a tool for semi automatically extracting structured and semistructureddata from text documents", SIGMOD Rec.27 (1998), 283–294.

[2] Z. Bar-Yossef and S.Rajagopalan, "Template Detection via Data Mining and Its Applications", *Proc.11th Int'l Conf.* World Wide Web(WWW), 2002.

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

[4] L.Yi, B. Liu and X. Li, "Eliminating noisy information in Web Pages for Data Mining", *In Proceedings of the International ACM Conference on Knowledge Discovery and Data Mining*, 2003

[5] L. Ma, N. Goharian, A. Chowdhury and M. Chung, "Extracting Unstructured Data from Template Generated Web Documents", *Proc. CIKM*, pp 512-515, 2003.

[6] T. Weninger, T., W.H.Hsu and J. Han, "CETR: content extraction via tag ratios", *Proceedings of the 19th International Conference on World Wide*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Web, *ACM*, New York, NY, USA, pp. 971–980, 2010.

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

[8] Liang Chen, Shaozhi Ye, Xing Li, "Template Detection for large scale search engines", *Proc.ACM Symposium*, pp. 1094-1098, 2006.

[9] Jushmerick. N, "Learning To Remove Internet Advertisements", *AGENT-99*, 1999.

[10] Yu Wang, Bingxing Fang, Xueqi Cheng, Li Guo and Hongvo Xu, "Incremental Web Page Template Detection", *Proc.17th Int'l Conf. World Wide Web*(WWW), pp. 1247-1248, 2008

[11] 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.

[12] Sandip Debnath, Prasenjit Mitra, C. Lee Giles, "Automatic Extraction of Informative Blocks from Web Pages", Proc. ACM Symposium, pp. 1722-1726, 2005.

[13] S. Zheng, D. Wu, R. Song, J-R. Wen,"Joint Optimization of Wrapper Generation and Template Detection", Proc. ACM SIGKDD, 2007.

[14] V. Crescenzi, P. Merialdo, P. Missier,"Clustering Web Pages Based on Their Structure", Data and Knowledge Eng., vol. 54, 2005, pp. 279-299.

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

[16] D. Gibson, K. Punera, A. Tomkins,"The Volume and Evolution of Web Page Templates", Proc. 14th Int'l Conf. World Wide Web (WWW), 2005.

[17] A. Kolcz, W. Yih,"Site-Independent Template Block Detection," Proc.KDD, Vol. 4702, 2007, pp. 152-163.

[18] Malcolm Slaney, Michael Casey,"Locality-Sensitive hashing for Finding Nearest Neighbors", IEEE Signal Processing magazine, March 2008.

[19] Y. Zhai, B. Liu,"Web Data Extraction Based on Partial Tree Alignment", Proc.14th Int'l Conf. World Wide Web(WWW), 2005.