ISSN No. 0976-5697

Volume 3, No. 4, July- August 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Enhancing Template Extraction accuracy of Heterogenous Web Documents

Mr.S.Sathees Babu
Department of Computer Science and Engineering
P.S.N.A. College of Engineering and Technology
Dindigul, Tamil Nadu, India
ssbabu@psnacet.edu.in

Abstract: Countless websites contain large set of pages generated using the common templates with contents. Due to the extraneous terms in templates, they degrade the accuracy and performance of web applications. Thus, template detection techniques have received a lot of attention recently to enhance the performance of web applications such as search engines, clustering, and classification. Thus, in order to prevent the duplication in the templates, nowadays we handle them with some detection techniques. In this paper, we present techniques for automatically cropping clusters based on MDL cost that can be used to extract search result records from dynamically generated web documents. Thus, we don't need additional template extraction process after clustering. Experimental results show that our proposed approach is feasible and effect for improving extraction accuracy.

***Keywords*:** Minimum Description Length (MDL), template extraction, MinHash, Max Algorithm, dice algorithm, clustering

I. INTRODUCTION

Internet is the source of information in recent decades. It helps us gaining more over everything all around the world. Some web pages are created based on some common template. Due to the irrelevant terms in the template, it degrades the performance of search engine. In this paper, we can detect and extract the common template from heterogeneous web pages. From this, it improves the performance of search engine, classification and clustering. Good template extraction technique can improve the performance of applications like industries, medical, Government and etc.

Extracting the common template from the homogeneous web documents has been studied in [2][3][4]. In this, the URL of the web documents is identical. All the documents are included in the same cluster when we use only URLs to group documents. To overcome the limitation of this problem, we can extract the templates from the heterogeneous web documents; the correctness of the extracted templates depends on the quality of clustering.

Extracting the template based on DOM tree presented in [2][5]. In this, it uses the tree edit distances measure for extracting common templates. However, it is not easy to select proper training data and not work for all the time. In [base paper], it employs Minimum Description Length principle for cluster the web documents and estimate the jaccard coefficient between sets. MDL cost of the clusters and execution time are high.

Our goal is to manage unknown number of templates and improve the efficiency and scalability of template detection and extraction algorithm. We extend the MinHash[7] by estimate the Dice's coefficient between two sets. It estimates MDL cost with partial information of documents. It is fully automated and robust without requiring many parameters. From the clustered document we can extract the data.

In summary, our contributions are as follows,

a. It constructs DOM tree for each document and calculate the support value for each path. It calculates the threshold value for each document. From this, it constructs essential path matrix.
b. We cluster the document based on the MDL Principle. It effectively manages unknown number of clusters. In this method, document clustering and template extraction are work together.
c. Experimental results confirm the effectiveness and scalability of our algorithm. The solution is much faster than related work and shows better accuracy.

We construct DOM tree for each document and calculate the support value for each path. We calculate the threshold value for each document. From this, we construct essential path matrix.

We cluster the document based on the MDL Principle. It effectively manages unknown number of clusters. In our method, document clustering and template extraction are work together. Experimental results confirm the effectiveness and scalability of our algorithm. Our solution is much faster than related work and shows better accuracy.

II. HTML DOCUMENTS AND DOCUMENT OBJECT MODEL

For example, let us consider simple HTML documents in Fig.1. We construct the Document Object Model (DOM) tree for the documents. The DOM presents an HTML document as a tree structure. For instance, the DOM tree of a simple document is shown Fig. 2. We find support values for each path in the documents based on the DOM tree. Document is represented as a set of paths {p1,p2,p3}.

```
<TABLE>
<TBODY>
<TR>
<TD>ShadyGrove</TD>
<TD>Aeolian</TD>
</TR>
</TBODY>
</TABLE>
```

Figure 1. Simple Web Document

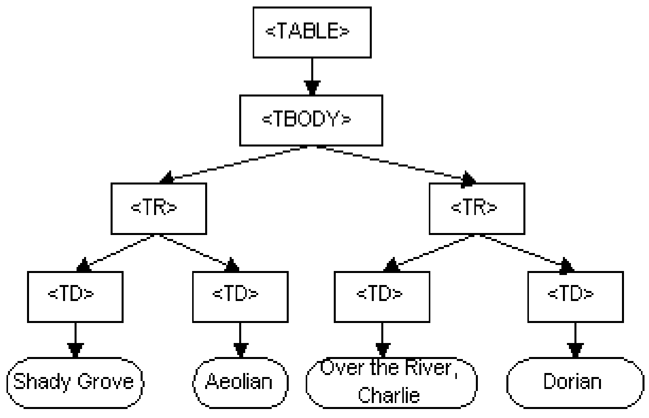
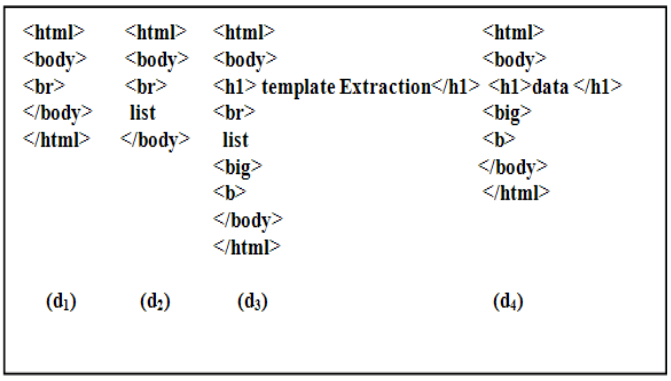


Figure 2. A graphical representation of the DOM figure1

For example, let us consider other sample HTML documents in Fig.3. We construct the Document Object Model (DOM) tree for the documents. The DOM presents an HTML document as a tree structure. We find support values for each path in the documents based on the DOM tree are shown in Table 1. Document d1 is represented as a set of paths {p1,p2,p3}. Document d2 is represented as a set of paths {p1,p2,p3,p4}.



```
<html>   <html>   <html>                          <html>
<body>   <body>   <body>                          <body>
<br>     <br>     <h1> template Extraction</h1>   <h1>data </h1>
</body>   list    <br>                            <big>
</html>  </body>   list                           <b>
                  <big>                           </body>
                  <b>                             </html>
                  </body>
                  </html>

 (d1)     (d2)     (d3)                            (d4)
```

Figure 3. Simple Web Documents

For a node in a DOM tree, we denote the path of the node by listing nodes from the root to the node in which we use ’\’ as a delimiter between nodes. For example, in the DOM tree of d3 in Figure 1, the path of a node ’Template Extraction’ is Document\<html>\<body>\<h1>\Template Extraction.

Table: 1 PATHS of tokens and their supports

	Path	Supports
P1	Document\<html>	4
P2	Document\<html>\<body>	4
P3	Document\<html>\<body>\<br>	3
P4	Document\<html>\<body>\list	2
P5	Document\<html>\<body>\<h1>	2
P6	Document\<html>\<body>\<big>	2
P7	Document\<html>\<body>\<b>	2
P8	Document\<html>\<body>\<h1>\ TemplateExtraction	1
P9	Document\<html>\<body>\<h1>\data	1

III. ESSENTIAL PATHS AND TEMPLATES

Collection of web documents can be represented by D = {d1,d2,…,dn}. We define a path set P_D as the set of all paths in D. For each document di, we calculate the threshold value. Threshold values for each document are distinct. The mode of each document is very effective to make templates, while contents are eliminated. We use the mode of support value for each document as the minimum support values of paths in each document as the minimum support threshold for each document. If several modes of support values, we will take the smallest mode. For example, In Fig. 1 and Table 1, the paths appearing at the documents d1 are p1, p2 and p3 whose supports are 4, 4, and 3 respectively. Since 4 is the mode of them, we use 4 as the minimum support threshold value for td1. Then p1 and p2 are the essential paths of d1. Similarly the minimum support thresholds td2, td3, and td4 are 4, 2, and 2. Based on the threshold value for each document we construct essential path matrix. Row represents paths in the document set and column represents documents. We use a | P_D | X |D| matrix ME with 0/1 values to represent the documents with their essential paths. If a path pi is an essential path of a document dj then the matrix ME is 1. Otherwise it is 0.

Example 1: Consider the HTML documents D = {d1, d2, d3, d4} in Figure 1. All the paths and their frequencies in D are given in Table 1. Assume that the minimum support thresholds td1, td2, td3, and td4 are 4, 4, 2 and 2 respectively. The essential path sets are E (d1) = {p1, p2}, E (d2) = {p1, p2}, E(d3) = {p1, p2, p4, p5, p6, p7} and E(d4) = {p1, p2, p4, p5, p6, p7}. We have the path set $P_D = \{pi|1 \le i \le 9\}$ and the matrix M_E becomes as follows:

$$ME = \begin{array}{c|cccc} & d1 & d2 & d3 & d4 \\ \hline P1 & 1 & 1 & 1 & 1 \\ P2 & 1 & 1 & 1 & 1 \\ P3 & 0 & 0 & 0 & 0 \\ P4 & 0 & 0 & 1 & 1 \\ P5 & 0 & 0 & 1 & 1 \\ P6 & 0 & 0 & 1 & 1 \\ P7 & 0 & 0 & 1 & 1 \\ P8 & 0 & 0 & 0 & 0 \\ P9 & 0 & 0 & 0 & 0 \end{array}$$

IV. FORMATION OF CLUSTERING BAESED ON MINIMUM DESCRIPTION LENGTH PRINIPLE

Initially each document is considered as a cluster. A cluster is denoted by a pair (T_i,D_i), where T_i is a set of essential path in the document and D_i is a set of documents in the cluster. Set of all the clusters are represented by C={c1,c2,…,cn} for a web document set D.(i.e) we have n clusters for a web document set D. Construct M_T, M_D and M_Δ matrix. M_T represents information of each cluster with its template paths and M_D represents the information of each cluster with its member documents. Construct M_Δ based on the formula $M_E = M_T.\ M_D + M_\Delta$. The MDL cost of the clustering and a matrix are denoted by L(C) and L (M), respectively. Cost of the clusters can be calculated by L(C) =L (M_T) + L (M_D) +L (M_Δ). L(M_D) becomes |D|.log2 |D|. L(M_T) and L(M_D) are calculated by

$$H(X) = \sum_{x\in\{1,0,-1\}} -\Pr(x)\log_2 \Pr(x)$$

$$L(M) = |M|\cdot H(X)$$

For example consider the web documents in Figure 1 and M_E in Example 1 again. Assume that we have a clustering C = {c1, c2} where c1 = ({p1, p2}, {d1, d2}) and c2 = ({p1, p2, p3, p4, p5, p6, p7}, {d3, d4}). Then, MT, MD and MΔ are as follows and we can see that $M_E = M_T \cdot M_D + M_\Delta$.

$$M_T = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad M_D = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$M\Delta = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Then, with M_T , Pr(1) = 9/36 and Pr(0) = 27/36 and we have $L(M_T) = |M_T| \cdot H(X) = 36 \cdot (- 9/36 \log_2 9/36 - 27/36 \log_2 27/36) = 29.21$. Similarly $L(M_D)=8$, $L(M_\Delta)=11.145$ and thus L(C)=48.35. In this way we can calculate MDL cost for different clusters combination and select minimum MDL cost cluster as a best cluster.

V. TEXT–MDL ALGORITHM USING AGGLOMERATIVE CLUSTERING ALGORITHM

For computation of Optimal cost calculation, we can reconstruct the formula [1] for calculating MDL cost as

$|ME| \cdot \beta / \alpha \cdot (\Pr(1)$ of $M_T + (\Pr(1)+\Pr(-1))$ of $M_\Delta) + L(M_D)$
$= \beta/\alpha \cdot (\#$ of 1s in $M_T + \#$ of 1s and -1s in $M_\Delta) + L(M_D)$ (1)

From this, cluster doesn't depend on the any cluster in the cluster set C.

```
procedure FindMDLCost($c_m$, $c_n$ , C)
begin
  a. $D_b := D_m \cup D_n$ ;
  b. $T_b := \{p_x | \sup(p_x, D_k) \geq |D_k|+1)/2 , p_x \in E_k\}$;
  c. $c_b := (T_b, D_b)$;
  d. $C'' := C - \{ c_m, c_n \} \cup \{ c_b \}$;
  e. MDL := Approximate MDL cost of C" by equation (1);
  f. return (MDL, $c_b$);
  End
procedure GetCorrectPair($c_b$, C)
begin
  a. $(c^A_m, c^A_n) :=$ the current best pair;
  b. $c^A_b :=$ a cluster made by merging $c^A_m$ and $c^A_n$;
  c. $MDL_{min} :=$ the current best approximate MDL cost;
  d. for each $c_l$ in C do {
  e.     $(MDL_{tmp}, c_{tmp}) :=$ GetMDLCost($c_m$, $c_n$ , C);
  f.     if $MDL_{tmp} < MDL_{min}$ then {
  g.         $MDL_{min} := MDL_{tmp}$;
  h.         $(c^A_m, c^A_n, c^A_b) := (c_b, c_l, c_{tmp})$;
  i.     }
  j. }
  k. return $(c^A_m, c^A_n, c^A_b)$;
  End
```

Initially get current best pair and merge the two clusters and calculate the MDL cost for the clusters. If this MDL cost is minimum compare to the initial MDL cost then we can merge the two clusters otherwise get another two clusters.

VI. TEXT – HASH ALGORITHM USING MINHASH FUNCTION

We will present the estimation of MDL cost of a clustering by MinHash. From that, we can reduce the dimensions of documents and find the best pair quickly.

A. Min Hash:

Jaccard's coefficient between two sets A1 and A2 is defined as $\gamma(A1, A2) = \frac{|A1 \cap A2|}{|A1 \cup A2|}$ and Dice's coefficient between two sets A1 and A2 is defined as $\partial(A1, A2) = \frac{2|A1 \cap A2|}{|A1|+|A2|}$. Both are calculated using Min-Wise Independent permutation. It estimates the coefficient by repeatedly assigning random ranks to the universal set and comparing the minimum values from the ranks of each set. For example refer[1].

Consider a set of random permutations $\Pi = \{\pi_1, \cdots, \pi_L\}$ on a universal set $U=\{r_1, \ldots r_M\}$ and a set $A1 \subset U$. Let π(ri) be the rank of ri in a permutation πi and min(πi(A1)) denote $\min(\pi_i(r_j) | r, \in A1)$. $\prod$ is called min- wise Independent if we have $\Pr(\min(\pi_i(A1) = \pi(x)) = 1/|A1|$ for every set $A1 \subset U$ and every $x \in A1$ for all $\pi_i \in \prod$ Then for any sets $A1, A2 \subset U$ for all $\pi_i \in \prod$,we have $\Pr(\min(\pi_i(A1)) = \min(\pi_i(A2))) = \gamma(A1, A2)$, where $\gamma(A1, A2)$ is the Jaccard's coefficient defined previously and $\partial(A1, A2)$ is the Dice's Coefficient.

For K sets $A_1, A_2, .., A_k$, the Jaccard's coefficient is defined by

$$\gamma(A_1 \ldots\ldots A_k) = \frac{|A_1 \cap \ldots\ldots\ldots \cap A_k|}{|A_1 U \ldots\ldots\ldots U A_k|}$$

For k sets A1, $A_2 \ldots\ldots$ Ak , the Dice's coefficient is defined by

$$\partial(A_1, ., A_k) = \frac{2 . |A_1 \cap \ldots\ldots\ldots\ldots \cap A_k|}{|A_1| + \ldots\ldots\ldots. + |A_k|}$$

We can estimate $\gamma(A_1 \ldots\ldots A_k)$ using the signature vectors as follows [5][6]

$$\gamma(A_1 \ldots A_k) = \frac{|\{i | sigA1[i] = \cdots = sigAk[i]\}|}{|\Pi|}$$

B. Extended MinHash:

Thus, given a collection of sets S={A1,…,Ak}, we extend MinHash to estimate the probabilities needed to compute the MDL Cost. We denote the probability as

$$\xi(X,m) = \frac{|\{ rj \mid rj \text{ is included in m number of sets in } X\}|}{|A1 \; U \ldots U \; Ak|}$$

Then, $\xi(X,m)$ is defined for $1\le m\le |S|$ and $\xi(S,|S|)$ is the same as the Jaccard's coefficient of sets in S.

We can estimate $\xi(X,m)$ with sigX as follows [1].

$$\xi(X,m) = \frac{|\{ i \mid n(sigX[i]) = m\}|}{|\Pi|} \qquad (2)$$

C. Calculation of MDL Cost using MinHash:

Computation of MDL cost using n(Di,k). Recall that sup(px,Di) is the number of documents in Di having the path px as an essential path. Let n(Di,k) represent the number of paths px whose sup(px,Di) is k. the following formula shows that we can count the numbers of 1s and -1s in MT and MΔ is follows

$$n(D_i,k) = \frac{\xi(D_i,k).\sum_{d_j\in D_i}|d_j|}{\sum_{1\le l\le |D_i|} l.\xi(D_i,l)} \qquad (3)$$

For proof refer [1];

For k with $1\le k\le |Di|$, we have

$$\sum_{1\le k<\frac{|D_i|+1}{2}} k.n(D_i,k) + \sum_{\frac{|D_i|+1}{2}\le k\le |D_i|} (|D_i|-k+1).n(D_i,k) \qquad (4)$$

D. Algorithm:

```
procedure FindHashMDLCost($c_m$, $c_n$ , C)
begin
a. $D_b := D_m U D_n$ , $c_b$ := (NULL, $D_b$), C'' := C– {$c_m$, $c_n$}U {$c_b$};
b. for each $\pi_q$ in $\Pi$ do {
c. r(sig$D_b$ [q]) := min(r(sig$D_m$ [q]), r(sig$D_n$ [q]));
d. if r(r(sig$D_m$ [q]) == r(sig$D_n$ [q]) then
e. n(sig$D_b$ [q]) := n(r(sig$D_m$ [q]) + n(sig$D_n$ [q]);
f. else n(sig$D_b$ [q]) is from the less one;
g. }
h. Calculate $\xi(D_b, l)$ by equation 2;
i. Compute n($D_b$, b) by equation4;
j. Get Pr(1) and Pr(−1) in $M_T$ and $M_\Delta$ by equation 3;
k. MDL := Approximate MDL cost of C'' by equation 1;
l. return (MDL, $c_b$);
   end
```

In this algorithm we estimate the MDL cost, but do not generate the template paths of each cluster. c_b is initialized as the empty set. We can use the signature of c_b is maintained to estimate the MDL cost.

VII. TEXT – MAX ALGORITHM USING MINHASH FUNCTION

When we merge the clusters hierarchically, we select two clusters which maximize the reduction of the MDL cost by merging them. In order to efficiently find the nearest cluster, we can calculate Jaccard's coefficient between two cluster's c_m and c_n as follows

$$\frac{\bigcup_{d_k\in(D_m\cup D_n)}E(d_k)}{\bigcap_{d_k\in(D_m\cup D_n)}E(d_k)}$$

Then, given three clusters c_m, c_n and c_k, if Jaccard's coefficient between cm and c_n is greater than that between cm and c_b, we assume that the reduction of the MDL cost by merging cm and c_n will be greater than that by c_m and c_b.

By using this approach, we can reduce the search space to find the nearest cluster. Using this approach, the search space becomes the number of clusters whose Jaccard's coefficient with cm is maximal. See the algorithm follows.

```
Procedure TakeInitBestPair(C)
begin
a. Merge all clusters with the same signature of MinHash;
b. $MDL_{min}$ := ∞;
c. for each cm in C do {
d. N := clusters with the maximal Jaccard's coeff. with cm;
   /* If the maximal Jaccard's coefficient is 0, N is NULL */
e. for each cn in N do {
f. ($MDL_{tmp}$ , $c_b$) := TakeHashMDLCost($c_m$, $c_n$ , C);
g. if $MDL_{tmp}$ < $MDL_{min}$ then {
h. $MDL_{min}$ := $MDL_{tmp}$;
i. ($c^A{}_m$, $c^A{}_n$, $c^A{}_b$) := ($c_m$, $c_n$, $c_b$);
j. }
k. }
l. }
m. return ($c^A{}_m$, $c^A{}_n$, $c^A{}_b$);
   end
```

```
Procedure TakeHashBestPair($c_k$, C)
begin
a. ($c^A{}_m$, $c^A{}_n$) := the current best pair;
b. $c^A{}_b$:= a cluster made by merging cAm and cAn;
c. $MDL_{min}$ := the current best approximate MDL cost;
d. N := clusters with the maximal Jaccard's coeff. with cb;
   /* If the maximal Jaccard's coefficient is 0, N is NULL */
e. for each cl in N do {
f. ($MDL_{tmp}$ , $c_{tmp}$) := FindHashMDLCost($c_b$, $c_l$, C);
g. if $MDL_{tmp}$ < $MDL_{min}$ then {
h. $MDL_{min}$ := $MDL_{tmp}$;
i. ($c^A{}_m$, $c^A{}_n$, $c^A{}_b$) := ($c_b$, $c_l$, $c_{tmp}$);
j. }
k. }
l. return ($c^A{}_m$, $c^A{}_n$, $c^A{}_b$);
   end
```

VIII. TEXT – DICE ALGORITHM USING MINHASH FUNCTION

In order to improve the efficiency for clustering we can implement another method Dice's coefficient calculation. Compare to previous algorithm it reduces the execution time and improve the MDL cost. Dice's coefficient between two cluster's c_m and c_n can be calculated as follows

$$\frac{2X\bigcup_{d_k\in(D_m\cup D_n)}E(d_k)}{\bigcap_{d_k\in(D_m)}E(d_k)+\bigcap_{d_k\in(D_n)}E(d_k)}$$

```
Procedure TakeDiceInitBestPair(C)
begin
a. Merge all clusters with the same signature of MinHash;
b. $MDL_{min}$ := ∞;
c. for each cm in C do {
d. N := clusters with the maximal Jaccard's coeff. with cm;
   /* If the maximal Dice's coefficient is 0, N is NULL */
e. for each cn in N do {
    f.  $(MDL_{tmp}, c_b)$ := TakeHashMDLCost($c_m$, $c_n$, C);
    g.  if $MDL_{tmp} < MDL_{min}$ then {
    h.      $MDL_{min} := MDL_{tmp}$;
    i.      $(c^A_m, c^A_n, c^A_b) := (c_m, c_n, c_b)$;
    j.  }
    k.  }
    l.  }
    m.  return $(c^A_m, c^A_n, c^A_b)$;
        end
Procedure **TakeDiceHashBestPair**($c_k$, C)
begin
    a.  $(c^A_m, c^A_n)$ := the current best pair;
    b.  $c^A_b$:= a cluster made by merging cAm and cAn;
    c.  $MDL_{min}$ := the current best approximate MDL cost;
    d.  N := clusters with the maximal Jaccard's coeff. with cb;
        /* If the maximal Dice's coefficient is 0, N is NULL */
    e.  for each cl in N do {
    f.      $(MDL_{tmp}, c_{tmp})$ := FindHashMDLCost($c_b$, $c_l$, C);
    g.      if $MDL_{tmp} < MDL_{min}$ then {
    h.          $MDL_{min} := MDL_{tmp}$;
    i.          $(c^A_m, c^A_n, c^A_b) := (c_b, c_l, c_{tmp})$;
    j.      }
    k.  }
    l.  return $(c^A_m, c^A_n, c^A_b)$;
        end
```

IX. EXPERIMENTAL RESULTS

A. Implemented Algorithms:

We implemented related work and our proposed algorithms as follows

a. ***TEXT-MDL:*** it is agglomerative clustering algorithm with the approximate entropy model.

b. ***TEXT-HASH:*** It is the agglomerative algorithm clustering algorithm with MinHash signature.

c. ***TEXT-MAX:*** It is the clustering algorithm with both MinHash signature and Jaccard's coefficient.

d. ***TEXT-DICEMAX:*** it is the clustering algorithm with both MinHash signature and Dice's coefficient.

B. Performance evaluation:

TEXT-MAX and TEXT-DICEMAX are faster execution time compare to TEXT-HASH without sacrificing accuracy. We compared the execution times and the MDL costs of TEXT-MDL, TEXT-HASH, TEXT-MAX and TEXT-DICEMAX with various numbers of documents from 1000 to 5000. Execution time is plotted in Fig 3a.

TEXT-DICEMAX, TEXT-HASH, TEXT-MDL and TEXT-MAX MDL cost are plotted in Fig 3b. TEXT-DICEMAX is having high MDL cost compare to other algorithm.

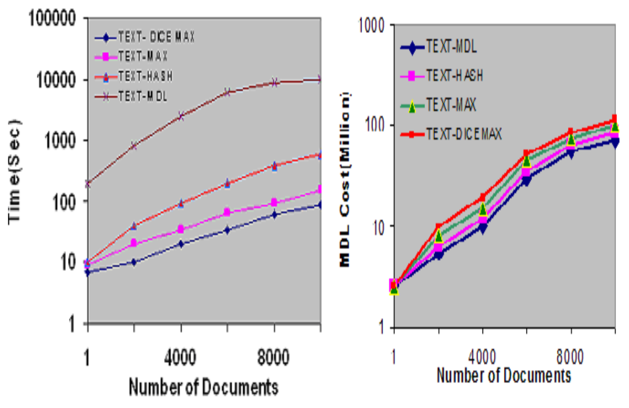


Figure: 4

X. CONCLUSION

We introduced a different algorithm based on MDL cost template detection from heterogeneous web documents. We employed the MDL principal to manage the unknown number of clusters. TEXT-DICEMAX algorithm shows the better execution time and high MDL cost value. It improves the performance of clustering and classification of template.

XI. REFERENCES

[1] Chulyun Kim and Kyuseok Shim,"Automatioc Template Extraction from Heterogeneous Web pages", IEEE Transactions on knowledge and data Engineering,vol 23,April 2011.

[2] 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,2004.

[3] I.S. Dhillon,S.Mallela, and D.S. Modha,"Information-Theoretic Co-Clustering', proc.ACM SIGKDD,2003.

[4] K.Vieira, A.S. da Silva ,N.Pinto, E.S. de Moura, J.M.B. Cavalcanti, and J.Freire, "A Fast and Roubust Method for Web Page Template Detection and Removal", Proc.15th ACM Int'l Conf. Information and Knowledge Management9cikm0,2006.

[5] S.Zheng, D. Wu, R.Song, and j.R.Wen, "Joint optimationa of wrapper Generation and Template Detection",proc. ACM SIGKDD,2007.

[6] A.Z. Broder, M.Charikar, A.M. Frieze, and M.Mitzenmacher, "Min- Wise Independent Permutations,",J.Computr and System Sciences,vol. 60,pp.630-659,2000.

[7] Z.Chen, F.Korn,N.Koudas, and S.Muithukrishnan, "Selectivity Estimation for Boolean Queries,",proc. ACM SIGMOD-SIGACT-SIGART Symp.Priciples of Database Systems(PODS),2000.

[8] A.Arasu and h.Garcia-Molina, "Etracting structured Data from Web Pages," Proc. ACM SIGMOD, 2003.

[9] Comparision of similarity coefficient based on RAPD markers in the common bean, http://dx.doi.org/10.1590/S1415-47571999000300024.