International Journal of Computer Applications (0975 – 8887)
Volume 119 – No.11, June 2015

Extraction of Template using Clustering from Heterogeneous Web Documents

Rashmi D Thakare
M.E Department of Computer Engineering
S.K.N College of Engineering
Pune, India

Manisha R Patil
Asst Prof, Department of Computer Engineering
S.K.N College of Engineering
Pune, India

ABSTRACT

In general, a common template or layout is used to generate set of pages in websites. For example, Google Book lays out the details like author name, book names, reviews or comments, etc. in the similar way in all of its book pages. The database provides different values to generate the pages. The problem during automatic database value extraction from different web pages is studied which is done without any human data input. A template is well defined which would propose the framework to be used to describe how the values are inserted into the pages. An extraction algorithm is at core to extract values from web pages. This algorithm is trained to generate the template referring defined set of words having common occurrence. As a result, extracted values are semantically similar in most of the cases. Ours focus on extracting templates from heterogeneous web pages. But due to large variety of web documents in websites, there is a need to manage unknown number of templates. This is achieved by clustering web documents. The various methods for clustering, which are compared i) TEXT Minimum Description Length (TEXTMDL), ii) MinHash using Jaccard Coefficient, iii) MinHash using Dice Coefficient methods are used for clustering web pages..

General Terms:

1st General Term, 2nd General Term

Keywords:

Webpage sectioning, webpage segmentation, template detection, Information extraction; Clustering; Web data modelling; Web data mining. Template Extraction, Data mining, Information search and retrieval.

1. INTRODUCTION

The information present on World Wide Web (WWW) is available in the form of structured data and unstructured data. Web readers easily access to contents by using templates. Nowadays, many people are interested in creation of their own websites, for this reason there have been several different ways to ease out the procedure of website making and its maintenance. Website template is an easy solution for making a website. Templates are coded in HTML format, which offer the use of the various different designs and graphics [1]. Web template reduces or eliminates professional web designers. Web template does not contain contents and photos. Any individual or company can use web templates to design and set up their website. After purchasing the template or downloading it, all its generic information are replaced with the user's personal, product or organizational information [2]. The unknown templates are degrading accuracy and performance due to irrelevant terms present in template. Nowadays, template detection and extraction have lot of attention. Fully automatic wrapper generation for search engines [3] presents method for automatically producing wrappers, which can be used to extract search results from dynamically generated result pages returned by search engines. This method utilizes visual content features on the result page and HTML tag structure of HTML source file. The problem of extracting templates which conform to common template has been studied in [4], [5], [6].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 application it is not feasible to crawl large number of documents and to classify them into homogeneous partitions in order to use these techniques. If we consider URL of web documents for grouping of web documents, then there may be different appearance of pages with same URL. Hence we cannot group web documents by using URL.

2. RELATED WORK

Template detection improves the performance of web application. Hence many researchers have tried to improve performance of template detection methodology.

Data extraction from HTML documents Document Object Model (DOM) tree can be used to represent HTML document, web documents are considered as trees. In Automatic Web News Extraction Using Tree Edit Distance [7], presents a domain oriented approach to web data extraction. This approach developed for finding and extracting data of interest from web pages. This approach is based on analysis of structure of these target pages. Here, they evaluate structural similarity between HTML pages and based on that, grouping of pages is done to form page cluster. Structure of web page described by a tree. Tree-edit distance is used to evaluate the structural similarity between pages. In Extracting structured data from web pages [4], extraction of data is done in two steps; I. Formally define a template and propose a model that describes how values are encoded into pages using a template II. Present algorithm that takes as input a set of template generated pages; deduce the unknown template used to generate pages and extracts as output.

Data extraction from XML documents XTract: A system for extracting Document Type Descriptor from XML Documents [9], provides a system for extracting Document Type Descriptor (DTD) schema from a database of XML documents. DTD contains valuable information on the structure of document. XTract method solved the problem of DTD extraction from multiple XML documents. MDL cost is used to find good DTD.

Clustering methods In Automatic Web News Extraction Using Tree Edit Distance [7], presents a method, in which small numbers of sampled documents are clustered first, and then the other documents are classified to the closest clusters. In this approach to select proper training data is difficult work. In Joint Optimization of Wrapper Generation and Template Detection [10], labelled training data is used for clustering. Each HTML document is mapped to Document Object Model (DOM) tree, then wrapper function (W) is defined for each DOM tree. Wrapper-Distance (WD) is used for Wrapper-oriented page clustering algorithm. The World Wide Web is a vast and rapidly growing source of information. Most of this information is in unstructured HTML pages that are targeted at a human audience. The unstructured nature of these pages makes it hard to do sophisticated querying over the information present in them. There are, however, many web sites that contain a large collection of pages that have more structure. Web pages use well defined sources like relational database to encode the data during run-time. The classic analysis of hashing schemes often entails the assumption that the hash functions used are random. More precisely, the assumption is that keys belonging to a universe U are hashed into a table of size M by choosing a function h uniformly at random among all the functions U! [M]. (The notation [M] stands for the set 0, . . . , M?1. This is slightly non-standard, but convenient for our purposes.) This assumption is impractical since just specifying such a function requires —U— log (M) bits1, which usually far exceeds the available storage [2]. Nowadays, web pages are generated with high content management systems; which gives high quality browser experience to the users. This consists of multiple navigation links, notices related to copyright and timestamp; this simplifies user information and access. There is side effect of these template structures which can disturb the web content by losing clarity of actual topic from the webpage. Further they also badly impact the effectiveness of search engine modules which includes index, duplicate spotting, summarization and ranking function. There is rapid and steady template contents growth [3, 11] leading to more than half of HTML present on the Web Pages. This makes it necessary for search engines tools/techniques to detect webpage templates more accurately. It is observed that most of template detection methods uses single website to analyze multiple webpages from the site and to identify repeated structures across the web sites. These template detection methods, which are operating at site-level though looking promising has limitations a First, less percentage of web templates are comprised of this site level template [3]. There are many web applications, which need reliable and quick estimation of number of strings which matches Boolean query. Ex: applications like alphanumeric query optimization, document counting with several queries. Here Boolean queries include substrings with which we can predicates composed using Boolean operators. Though some work for selective substring query has been conducted, the problem for selective Boolean query estimation has not been well studied [4].The technique that the paper looked into to extract data from HTML uses wrappers, generated automatically. In this paper, we develop novel techniques which is used to compares HTML pages and build up wrapper on the basis of differences and similarities. When we conducted experiments on web-sties which are data intensive with real-life example, the results are encouraging to implement this approach [5]. There is large number of web sites which has structured data. The web pages from these regions get automatically generated with the help of programs which extracts data from back end database and then implant into an HTML template. This results into pages which are generated by the same program showing common layout and structure, at the same time differing in content [6]. Clustering is one of the tool in unsupervised learning which is used to bunch similar data types. This has practical importance in applications like data analysis based on web-log, text and market-base.

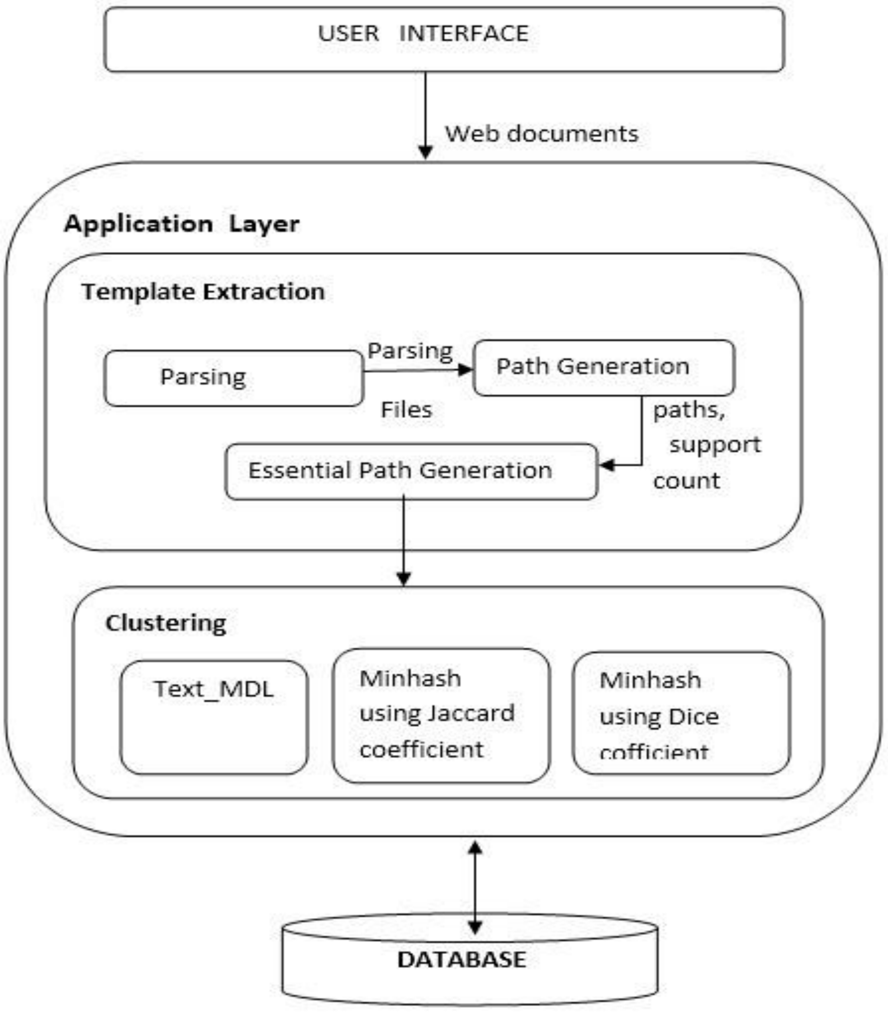
3. PROPOSED APPROACH FRAMEWORK AND DESIGN

3.1Problem Definition

Currently the way of extracting the web documents are not that much advanced to extract the web documents from heterogeneous templates. Clustering is required for each and every type of Template. But the quality of extracted templates truly depends upon the type of clustering used on it. As it is based on the URL identification, the result is not that much Good or Effective. So, we present the novel algorithm to extract template of web document from vast number of web documents which are generated from heterogeneous templates. The web documents are clustered depending upon underlying template structures similarity in the documents so that each cluster template is further extracted simultaneously.

3.2Proposed Architecture and Design

The data flow architecture for template extraction system is shown in fig. 1. Here three layers present; user layer, application layer and database layer. At user layer, user interacts with template extraction system. Template extraction and clustering processes are present at application layer. At the database layer paths, support count, clusters, execution time, MDL cost are stored. Template extraction system takes input only in the form of HTML. These HTML pages parsed using HTML parser. System finds the paths or flow of document structure of HTML pages by using its tag entry, and system also finds support count of paths. HTML document path and support count of path generate during template extraction process, which store at database. Path and support count of path are used to find essential path of documents. To manage number of templates we have clustered HTML pages based on template structure. We have used three clustering methods TEXTMDL, MinHash using Jaccard Coefficient, MinHash using Dice coefficient for clustering of HTML documents. HTML documents presents in cluster have same template paths. Each cluster extracts their templates simultaneously. Outputs of system are template paths of HTML pages and clusters of HTML pages. HTML document can be represented by Document Object Model (DOM) tree. DOM is easy way to represent structured information of HTML document. DOM is application programming interface for valid HTML. It defines logical structure of document. DOM is platform and language neutral interface that w document content, structures and style to be dynamically updated by the program scripts. The HTML document, HTML element, texts in the HTML document, HTML attribute and comments are represented in DOM tree as document node, an element node, text nodes, attribute node and comment node respectively.



A. Template Extraction System

Template extraction method consists of following steps as we men tioned in : HTML document and Document Object Model (DOM) tree Paths of HTML documents Support count of paths Essential paths of documents Cluster HTML documents based on template of HTML documents using TEXTMDL [12], MinHash Jaccard Coefficient [12], and MinHash Dice Coefficient.

B. HTML Document and Document Object Model

HTML document can be represented by Document Object Model (DOM) tree. DOM is easy way to represent structured information of HTML document. DOM is application programming interface for valid HTML. It defines logical structure of document. DOM is platform and language neutral interface w document content, structures and style to be dynamically updated by the program scripts. The HTML document, HTML element, texts in the HTML document, HTML attribute and comments are represented in DOM tree as document node, an element node, text nodes, attribute node and comment node respectively. We have studied all HTML tags using Ref. [13].

C. Essential paths of document

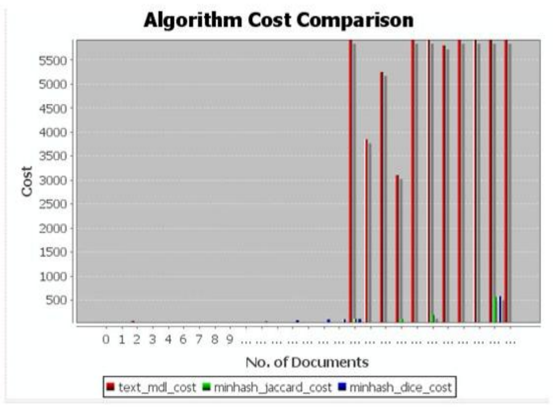
Define path set PW for document W. PW is set of all paths in W. Support of path(SP) is the number of documents in W, which contain the path. For all wi, exists twi. The twi is decided by taking mode of C. values. If a path is contained by a document wi and support of path is at least the given twi then the path is essential path of wi. Set of such essential paths are EP. These essential paths are used in extracting template. Set of paths and HTML documents are denoted by matrix of size —PW——W— and that matrix is denoted as ME.

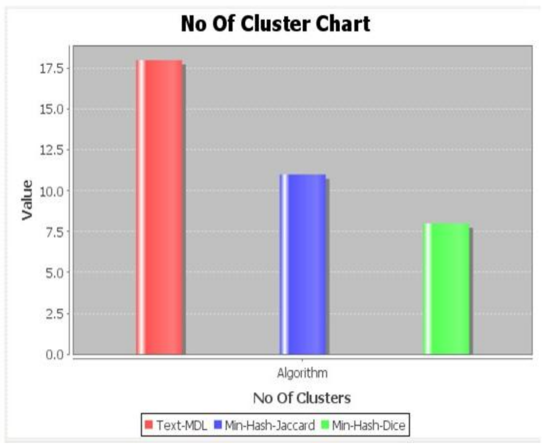
D. Clustering of HTML documents

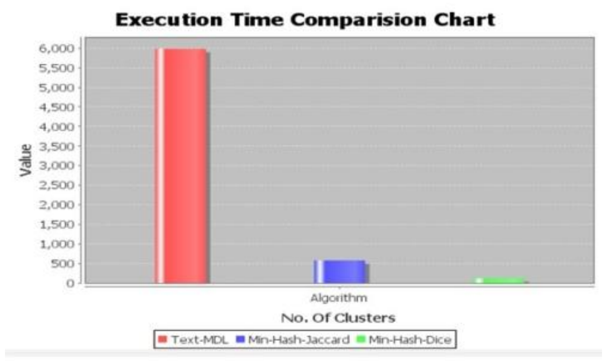
We have done clustering of HTML documents using three methods:
i) TEXTMDL[12], ii) MinHash using Jaccard Coefficient[12], iii) MinHash using Dice Coefficient.

4. WORK DONE

Our goal is to manage an unknown number of templates and further enhance the scalability and exact efficiency of template detection and extraction algorithms. To deal with the unknown number of templates and select good partitioning from all possible partitions of web documents, we employ Rissanen?s Minimum Description Length (MDL) principle in. Intuitively, each candidate partitioning (i.e., clustering) is ranked according to the number of bits required to describe a clustering model and the partitioning with the minimum number of bits is selected as the best one. In our problem, after clustering documents based on the MDL principle, the model of each cluster is the template itself of the web documents belonging to the cluster. Thus, we do not need additional template extraction process after clustering. In order to improve efficiency and scalability to handle a large number of web documents for clustering, we extend MinHash. While the traditional MinHash is used to estimate the Jaccard coefficient between sets, we propose an extended MinHash to estimate our MDL cost measure with partial information of documents. Also, algorithms that we proposed are robust and fully automated that doesn't need many parameters. Actual experimental results real life data confirm our algorithm effectiveness. In summary, our contributions are as follows: We apply the MDL principle to our problem to effectively manage an unknown number of clusters (i.e., an unknown number of templates). In our method, document clustering and template extraction are done together at once. The MDL cost is the number of bits required to describe data with a model and the model in our problem is the description of clusters represented by templates. Since a large number of web documents are massively crawled from the web, the scalability of template extraction algorithms is very important to be used practically. Thus, we extend MinHash technique to estimate the MDL cost quickly, so that a large number of documents can be processed. Experimental results with real life data sets up to15 GB confirmed the effectiveness and scalability of our algorithms. Our solution is much faster than related work and shows significantly better accuracy.







With our experiments, results are as follows: Algorithm Cost Comparison: Y-axis indicates cost in million while X-axis is number of documents searched TextMDL is costlier as compared to MinHash Jaccard and MinHash Dice algorithms. MinHash Dice is most cost effective algorithm. Cluster Chart Comparison: Y-axis indicates number of cluster generated MinHashDice gives minimum no of clusters as compared to others. This results in higher accuracy during search. Execution Time Comparison: Y-axis indicates time in msec for cluster generation Again, MinHashDice is very faster during cluster generation

5.MATHEMATICAL MODEL

Input is given as a Web Documents
Doc, Doc path, E path Where, Doc is a set of Web Documents.
Doc path is a set of Document Paths. E path is a set of Essential Paths.
Output is given as a Clusters
C1 , C2, C3
Where, C1 is a set of Clusters by TextMDL.
C2 is a set of Clusters by MinHash.
C3 is a set of Clusters by Minhash with Dice.
Process is given as a Clusters P1 ,P2,P3
Where, P1 = {e1, e2, e3, e4, e5} set of processes for pre-processing Where,{e1=$i|i$ is to generate the Dom tree}
{e2=$j|j$ is to generate the paths}
{e3=$k|k$ is to find the support count for each path}
{e4=$l|l$ is to generate the essential paths from support count}
{e5=$l|l$ is to generate the essential paths}
P2 = {e1, e2, e3, e4} be the set of processes for Clustering by TextMDL
Where, {e1=$i|i$ is to form the matrices ME, MD, MT} MT = information of each cluster with its template paths. MD = denotes the information of each cluster with its member documents. M_Δ = is a difference matrix.

$$ME = MT * MD + M_\Delta$$

{e2=$j|j$ is to calculate MDL cost for each cluster pairs}
Probability of 1 in MT = count of 1 in MT / Total Element in MT Probability of 0 in MT = 1 - Probability of 1 in MT

$$LMT= Total\ ELE\ MT * \Sigma_{x\in\{1,0,-1\}} - P_r(x) log_2 P_r(x)$$

Find same for $LM\Delta$

LMD=Total_files * log(totalFiles) / log(2)

$$MDL = LMD + LM_\Delta + LMT$$

{e3=$k|k$ is to get the best pair according to MDL cost}
{e4=$l|l$ is to form Clusters}

P3 = {e1, e2, e3, e4,} be the set of processes for Clustering by MinHash with Jaccard Coeff
Where,{e1=$j|j$ is to calculate Permutations and signatures values for each essential path} $sig_{s1} = [min(\pi_1(S1)),min(\pi_2(S1)),.....,min(\pi_L(S1))]$

Jaccard coeff is calculated by

$$\varepsilon(S1,\dots\dots,Sk) = \frac{|S1 \cap \dots\dots.\cap Sk|}{|S1 \cup \dots\dots.\cup Sk|}$$

{e2=$j|j$ is to calculate MDL cost for each cluster pairs}
$|M\epsilon|\frac{\beta}{\alpha}(\Pr(1) Of\ M\tau + (\Pr(1) + \Pr(-1))\ of\ M\Delta) + L\ (M_D) = \frac{\beta}{\alpha}(\#\ of\ 1s\ in\ M\tau + \#\ of\ 1s\ and - 1s\ in\ M\Delta) + L\ (M_D)$
{e3=$k|k$ is to get the best pair according to MDL cost}
{e4=$l|l$ is to form Clusters}

Algorithm
System S, for template extraction can be defined as
S = {W, T, C, DOM, PW, SP, tw, EP, ME, CM, MDL, MT, MD,
Mdelta}
W is set of web document, W= {w1, w2, w3, ...wm}.
T is set of template paths, T= {t1, t2, t3, ...tn}.
C is set of cluster, C= {c1, c2, c3 ... ck}.
DOM is set of Document Object Model tree, DOM= {d1, d2, d3 ...dm}.
PW is set of all paths in W. PW = {p1, p2, p3... pk}. SP is support of path, which is the number of documents in W, which contains the path.
tw is minimum support threshold. tw = mode of SP
EP is set of essential paths. EP = {ep1, ep2,...,epl}
ME matrix denotes web documents with its essential paths; its size is $|PW|x|W|$
CM is set of clustering methods.
CM= {TEXTMDL, MinHash Jaccard Coefficient, MinHash Dice
Coefficient}
MDL is Minimum Description Length principle.
MT matrix represents information of each cluster with its template path,
MT is of size $|PW||W|$
MD is matrix which represents information of each cluster with its member documents;
MD is of size $|D||D|$
Mdelta is a difference matrix.

TEXTMDL
Input: W, Ep
Output: C Process:

1$Optimal template paths T_i are given as, T_i \in EP and T_i have SP \geq ((|Wi + 1)/2)$
2 $If pi \in T_j then MT(i,j) = 1 else MT(i,j) = 0$ 3 $If dj \in ci then MD(i,j) = 1 else MD(i,j) = 0$ 4 ME is given by formula: ME = MT * MD + Mdelta
Where Mdelta contains 0/1/-1.
5 MDL cost of clustering model C is given by formula:
L(C) = L(MT) +L(MD) +L(Mdelta).
L(MT) = $|MT|$*H(X),
L(MD) = $|MD|$*H(X),
L(Mdelta) = $|Mdelta|$*H(X), Where
$H(X) = \Sigma_{x\in\{1,0,-1\}} - P_r(x) log_2 P_r(x)$is the probability of x.
6 $If L(C) \leq L(C^0)$ then C is best clustering model than C'. 7 The Optimal MDL cost is given by formula,
L(C) = $|ME| * alpha/beta(Pr(1) of MT + (Pr(1) + Pr(-1)) of Mdelta) + L(MD)$,

Where alpha = Pr(1) in ME that means probability of 1 in matrix ME, beta = H(X) of ME

Web documents and essential paths are inputs to the TEXTMDL clustering algorithm. We find out ME matrix which contains essential paths and web documents. Initially each web document considered as cluster itself. MD matrix contains clusters along with its web documents. These MD matrixes generate with all possible combination clusters with its web documents. MT matrix contains clusters with its template paths. Template paths are essential paths but having support count greater than or equal to ((—Wi+1)/2). Then Mdelta matrix calculated using formula as mentioned in step 4. Then Minimum Description Length(MDL) cost of clustering model is find out using formula as mentioned in step 5. As MD matrix generated using all combination of clusters along with its web documents. Because of MD matrix the cost of clustering models differs from each other. We kept record of MDL cost of all clustering models then we have compared these costs and find out best clustering model with minimum MDL cost.

Method 2: MinHashJaccard Coefficient

Input: W, Ep

Output: C

Process:

1. Initially each web document is considered as cluster
2. Sig. is signature of document
3. If Sig(c1) = Sig(c2) then merge c1, c2 in one cluster.
4. Jaccard Coefficient (c1, c2) is given by formula:
$JC(c1,c2) = |c1 \cap c2|/|c1 \cup c2|$
5. Find cluster pair having maximal JC
6. Find MDL cost of clustering model
7. Find best clusters

MinHash using Jaccard Coefficient clustering algorithm takes input as web documents and essential paths.Initially each document is considered as cluster. We find signature values for each cluster as mentioned in [12]. We merge clusters having same signature values. We find out Jaccard Coefficient between clusters. We select cluster pair having maximal Jaccard Coefficient. Then we compute MDL cost for that clustering model. We compare MDL cost of all clustering models then we select clustering model with minimum MDL cost.

Method 3: MinHashDice Coefficient
Input: W, Ep

Output: C Process:

1. Initially each web document is considered as cluster
2. Sig is signature of document
3. If Sig(c1)= Sig(c2) then merge c1, c2 in one cluster
4. Dice Coefficient(c1, c2) is given by formula: $DC(c1,c2) = 2 * |c1 \cap c2|/|c1| + |c2|$
5. Find cluster pair having maximal DC
6. Find MDL cost of clustering model
7. Find best clusters

MinHash using Dice Coefficient clustering algorithm works similar as MinHash using Jaccard Coefficient algorithm. Only difference is at step 4, here we calculate Dice Coefficient (DC) between clusters. We have implemented MinHash Dice Coefficient algorithm which takes less execution time than MinHash Jaccard Coefficient. In results and discussion we compared clustering result of these three clustering algorithm. MinHash using Dice Coefficient clustering algorithm has better performance over MinHash using Jaccard Coefficient clustering algorithm.

6. RESULTS AND DISCUSSION

Template extraction method consist of different steps such as, taking input as HTML documents, creating Document Object Model (DOM) tree, finding paths of HTML documents, finding support count of paths, finding essential paths of documents, Clustering HTML documents based on template of HTML documents. For testing all these steps we have taken some input web documents that are shown in fig. 2. We have taken four web documents w1, w2, w3, and w4 as inputs.

A. Document Object Model (DOM) tree

Fig. 3 shows DOM tree d1 for web document w1. Fig. 4 shows DOM tree d2 for web document w2. Similarly we can build Document Object Model (DOM) tree for other web documents.

a.W1

```
<html>
<head>
<title >UMBC Agent Web</title>
</head>
<body>
<h1>A page from the UMBC AgentWeb</h1>
<p>Paragraph</p>
</body>
</html>
```

b.W2

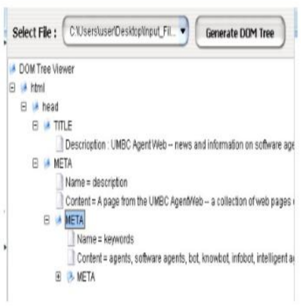
```
<html>
<head>
<title>Search</title>
</head>
<body>
<h1>Lesson</h1>
<p>Paragraph</p>
</body>
</html>
```

c.W3

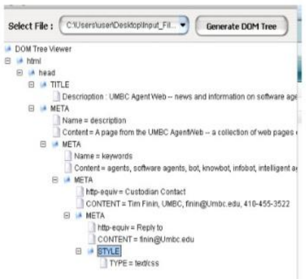
```
<html>
<head>
</head>
<body>
<h1>Header</h1>
</body>
</html>
```

d.W4

```
<html>
<head>
</head>
<body>
<table>
        <tr>
        <th>title</th>
        <th>price</th>
        </tr>
</table>
</body>
</html>
```



```
DOM Tree Viewer
⊟ html
    ⊟ head
        ⊟ TITLE
              Description : UMBC Agent Web -- news and information on software age
        ⊟ META
              Name = description
              Content = A page from the UMBC AgentWeb -- a collection of web pages
            ⊟ META
                  Name = keywords
                  Content = agents, software agents, bot, knowbot, infobot, intelligent a
                ⊞ META
```



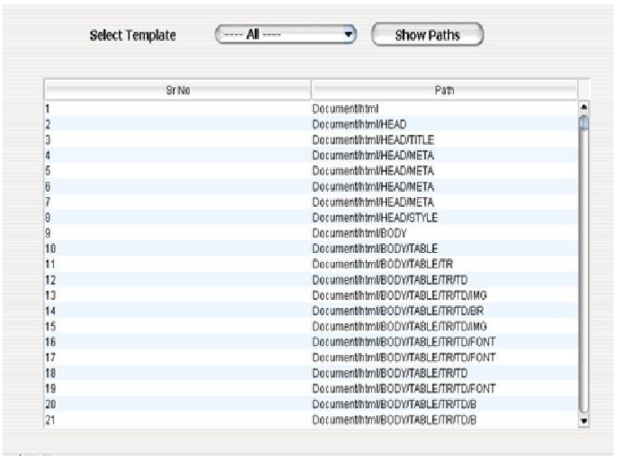
```
DOM Tree Viewer
⊟ html
    ⊟ head
        ⊟ TITLE
              Description : UMBC Agent Web -- news and information on software age
        ⊟ META
              Name = description
              Content = A page from the UMBC AgentWeb -- a collection of web pages
            ⊟ META
                  Name = keywords
                  Content = agents, software agents, bot, knowbot, infobot, intelligent a
                ⊟ META
                      http-equiv = Custodian Contact
                      CONTENT = Tim Finin, UMBC, finin@Umbc.edu, 410-455-3522
                    ⊟ META
                          http-equiv = Reply to
                          CONTENT = finin@Umbc.edu
                        ⊟ STYLE
                              TYPE = text/css
```

Fig.3 DOM tree d for web document W

B. Paths of HTML document

Paths presents in web document W are shown in Table I. Similarly we can find paths present in other web documents.

Table I Paths Representation



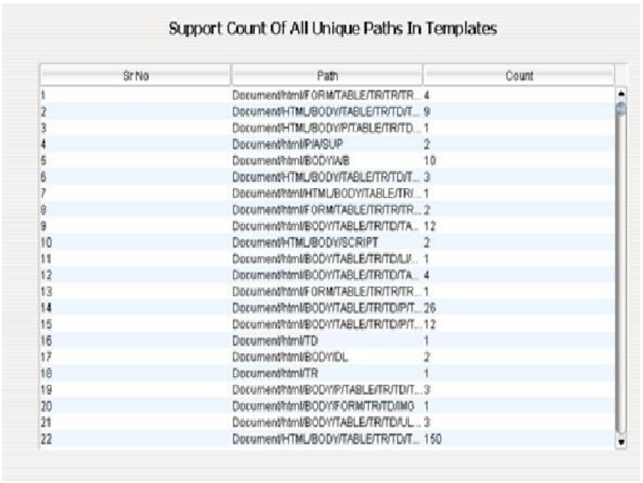
Sr No	Path
1	Document/html
2	Document/html/HEAD
3	Document/html/HEAD/TITLE
4	Document/html/HEAD/META
5	Document/html/HEAD/META
6	Document/html/HEAD/META
7	Document/html/HEAD/META
8	Document/html/HEAD/STYLE
9	Document/html/BODY
10	Document/html/BODY/TABLE
11	Document/html/BODY/TABLE/TR
12	Document/html/BODY/TABLE/TR/TD
13	Document/html/BODY/TABLE/TR/TD/IMG
14	Document/html/BODY/TABLE/TR/TD/BR
15	Document/html/BODY/TABLE/TR/TD/IMG
16	Document/html/BODY/TABLE/TR/TD/FONT
17	Document/html/BODY/TABLE/TR/TD/FONT
18	Document/html/BODY/TABLE/TR/TD
19	Document/html/BODY/TABLE/TR/TD/FONT
20	Document/html/BODY/TABLE/TR/TD/B
21	Document/html/BODY/TABLE/TR/TD/B

fig 4: Paths represent in W

C. Support count of paths

Support count of all paths present in web document w1, w2, w3 and w4 is shown in Table III. Path ”Document/html” present in w1, w2, w3 and w4, so its support count path ”Document/html/head/title/” is present only in w1, so its we can count support count of web documents also we can count same for other paths we calculate support count.

Table III. Support count of all unique paths



Sr No	Path	Count
1	Document/html/FORM/TABLE/TR/TR/TR...	4
2	Document/HTML/BODY/TABLE/TR/TD/T...	9
3	Document/HTML/BODY/P/TABLE/TR/TD...	1
4	Document/html/P/A/SUP	2
5	Document/html/BODY/A/B	10
6	Document/HTML/BODY/TABLE/TR/TD/T...	3
7	Document/html/HTML/BODY/TABLE/TR/...	1
8	Document/html/FORM/TABLE/TR/TR/TR...	2
9	Document/html/BODY/TABLE/TR/TD/TA...	12
10	Document/HTML/BODY/SCRIPT	2
11	Document/html/BODY/TABLE/TR/TD/LI/...	1
12	Document/html/BODY/TABLE/TR/TD/TA...	4
13	Document/html/FORM/TABLE/TR/TR/TR...	1
14	Document/html/BODY/TABLE/TR/TD/P/T...	26
15	Document/html/BODY/TABLE/TR/TD/P/T...	12
16	Document/html/TD	1
17	Document/html/BODY/DL	2
18	Document/html/TR	1
19	Document/html/BODY/P/TABLE/TR/TD/T...	3
20	Document/html/BODY/FORM/TR/TD/IMG	1
21	Document/html/BODY/TABLE/TR/TD/UL...	3
22	Document/HTML/BODY/TABLE/TR/TD/T...	150

The above diagram show support count of all unique path in templates. Web pages are the HTML documents, to match the similarity we first pre-process the html document, to pre-process, parse the html document using DOM model, then generate path, calculate support count and then form the essential path. Now these essential paths are the input for the clustering.

Sr No:794 Path: Document*/html/FORM/TABLE/TR/TR /TR/TR/TR/TR/TR/TR/TR/TR/TR/TD/INPUT*
Support Count: 1

Sr No:795 Path: Document*/html/FORM/TABLE/TR/TR /TR/TR/TR/TR/TR/TR/
TR/TR/TR/TR/TR/TR/TR/TR/TR/TR/TR/TR/TD*

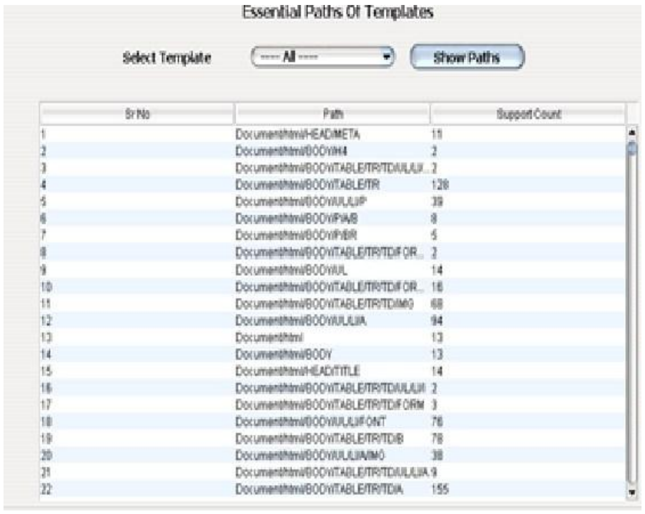
Support
Count: 4
Sr No:796
Path:
Document*/html/BODY/TABLE/TR/TD/P/TABLE/TR*
/TD
Support Count: 50

D. Essential paths of HTML documents

Essential paths of all web documents W are shown in Table IV. The Paths present in web documents W are shown in Table I. Paths along with their support count are shown in Table III. Paths present in w1 are ”Document/html”, ”Docu-ment/html/head”,Document/html/head/title”,”Document/html/head/ title/Search”, ”Document/html/body”,”Document/html/body/h1”,”Document/html/ body/h1/Lesson”,”Document/html/body/p”, ”Document/html/body/p/Paragraph” .

Essential path for W are set of paths having support count greater than equal to 1.Paths present in w2 are ”Document/html”, ”Document/html/head”, ”Document/html/body”, ”Document/html/body/p”, ”Document/html/body/p/First Paragraph” and their support count are respectively. Hence ”Document/html”,”Document/html/head”,”Document/html/body” are essential paths of web document w2.Essential paths of all web documents W Table VI.

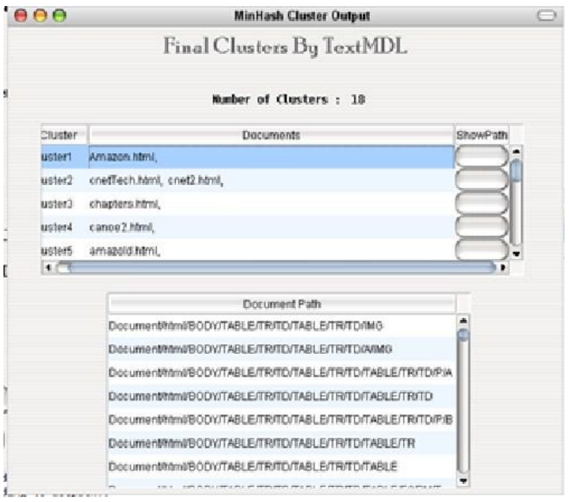
Table IV: Essential Paths of W


Essential Paths Of Templates

Select Template: ---- All ----

Sr No	Path	Support Count
1	Document/html/HEAD/META	11
2	Document/html/BODY/H4	2
3	Document/html/BODY/TABLE/TR/TD/UL/LI/...	2
4	Document/html/BODY/TABLE/TR	128
5	Document/html/BODY/UL/LI/P	39
6	Document/html/BODY/P/A/B	8
7	Document/html/BODY/P/BR	5
8	Document/html/BODY/TABLE/TR/TD/FOR...	2
9	Document/html/BODY/UL	14
10	Document/html/BODY/TABLE/TR/TD/FOR...	16
11	Document/html/BODY/TABLE/TR/TD/IMG	68
12	Document/html/BODY/UL/LI/A	94
13	Document/html	13
14	Document/html/BODY	13
15	Document/html/HEAD/TITLE	14
16	Document/html/BODY/TABLE/TR/TD/UL/LI/	2
17	Document/html/BODY/TABLE/TR/TD/FORM	3
18	Document/html/BODY/UL/LI/FONT	76
19	Document/html/BODY/TABLE/TR/TD/B	78
20	Document/html/BODY/UL/LI/A/IMG	38
21	Document/html/BODY/TABLE/TR/TD/UL/LI/A	9
22	Document/html/BODY/TABLE/TR/TD/A	155

Clustering Result Using TextMDL Method



Final Clusters By TextMDL

Number of Clusters : 18

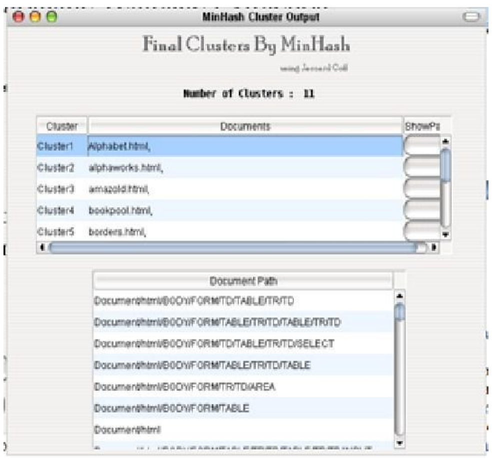
Cluster	Documents	ShowPath
uster1	Amazon.html,	
uster2	cnetTech.html, cnet2.html,	
uster3	chapters.html,	
uster4	canoe2.html,	
uster5	amazold.html,	

Document Path
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/IMG
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/A/IMG
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/TABLE/TR/TD/P/A
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/TABLE/TR/TD
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/TABLE/TR/TD/P/B
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/TABLE/TR
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/TABLE

As shown in fig.TEXT-MDL is an agglomerative hierarchical clustering algorithm which starts with each input document as an individual cluster. When a pair of clusters is merged, the MDL cost of the clustering model can be reduced or increased. The procedure GetBestPair finds a pair of clusters whose reduction of the MDL cost is maximal in each step of merging and the pair is repeatedly merged until any reduction is not possible.

```
P389 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
P390 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
P391 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
P392 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
P393 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
P394 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
P395 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
P396 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
P397 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
P398 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
***************************
Average MDL Cost For TextMDL : 6249.217108672593
19 inserting into output
19 pdating into output
```

Clustering Result Using MinHash Jaccard Coefficient Method



Final Clusters By MinHash

using Jaccard Coff

Number of Clusters : 11

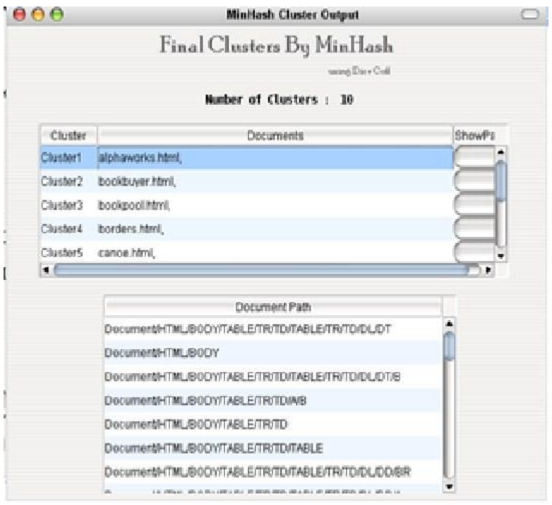
Cluster	Documents	ShowPs
Cluster1	Alphabet.html,	
Cluster2	alphaworks.html,	
Cluster3	amazold.html,	
Cluster4	bookpool.html,	
Cluster5	borders.html,	

Document Path
Document/html/BODY/FORM/TD/TABLE/TR/TD
Document/html/BODY/FORM/TABLE/TR/TD/TABLE/TR/TD
Document/html/BODY/FORM/TD/TABLE/TR/TD/SELECT
Document/html/BODY/FORM/TABLE/TR/TD/TABLE
Document/html/BODY/FORM/TR/TD/AREA
Document/html/BODY/FORM/TABLE
Document/html

MinHash using Jaccard Coefficient clustering algorithm takes input as web documents and essential paths. Initially each document is considered as cluster. We find signature values for each cluster .We merge clusters having same signature values.

We find out Jaccard Coefficient between clusters. We select cluster pair having maximal Jaccard Coefficient. Then we compute MDL cost for that clustering model. We compare MDL cost of all clustering models then we select clustering model with minimum MDL cost.
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/FOR
M/TABLE
Document/html/BODY/TABLE/TR/TD/FORM/IMG
Document/html/BODY/P/TABLE/TR/TD/TABLE/TR/TD/IMG
Document/html/BODY/TABLE/TR/TD/P/TABLE/TR/TD
Document/html/BODY/DL/DD
Document/html/BODY/TD/IMG
Document/html/BODY/TR
Document/html/BODY/DL/FONT
Document/html/BODY/P/TD
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/TABL
E
/TR/TD/A/IMG
Document/html/BODY/TABLE/TR/TD/P/FONT
Time required for template extraction using Jaccard coefficient: 685
Average MDL Cost For MinHash with Jaccard:
110.77466522385966
Pre=0.9728325656974354
recall=0.8227382907341814

Clustering Result Using MinHash Dice Coefficient Method

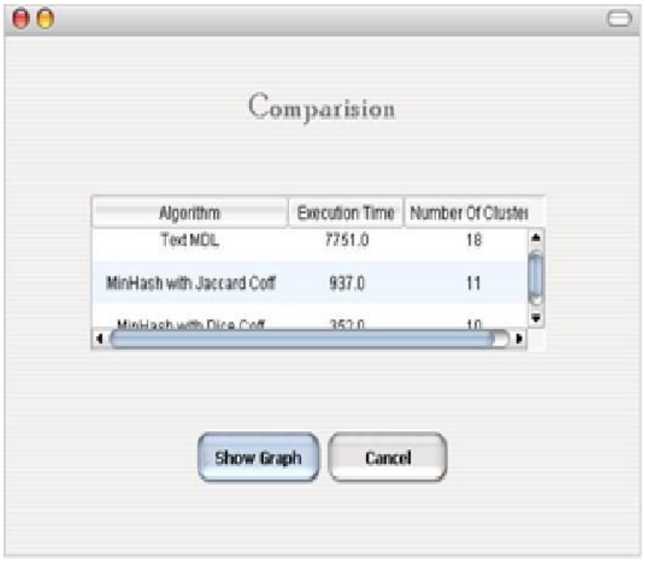


Cluster	Documents	ShowPs
Cluster1	alphaworks.html,	
Cluster2	bookbuyer.html,	
Cluster3	bookpool.html,	
Cluster4	borders.html,	
Cluster5	canoe.html,	

Document Path
Document/HTML/BODY/TABLE/TR/TD/TABLE/TR/TD/DL/DT
Document/HTML/BODY
Document/HTML/BODY/TABLE/TR/TD/TABLE/TR/TD/DL/DT/B
Document/HTML/BODY/TABLE/TR/TD/A/B
Document/HTML/BODY/TABLE/TR/TD
Document/HTML/BODY/TABLE/TR/TD/TABLE
Document/HTML/BODY/TABLE/TR/TD/TABLE/TR/TD/DL/DD/BR

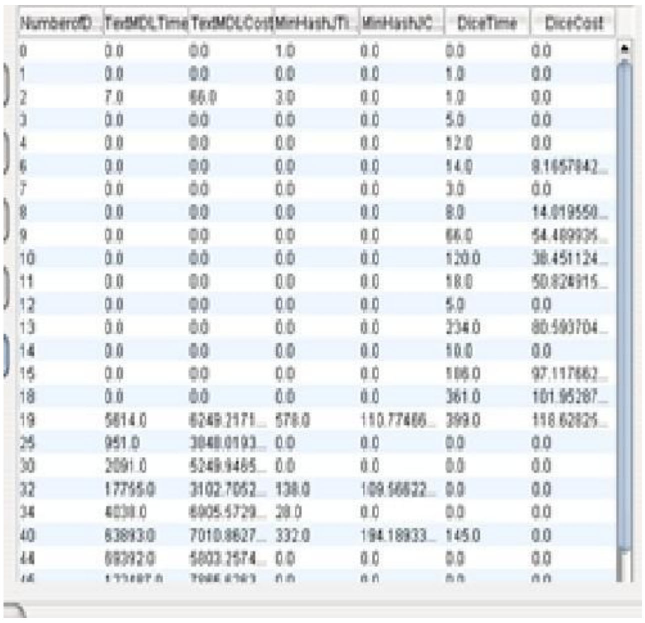
Document/html/BODY/TABLE/TR/TD/P/TABLE/TR/TD/TA
BLE/TR/
TD/FORM
Document/html/BODY/TABLE/TR/TD/TABLE/TR/FORM
Document/html/BODY/TABLE/TR/TD/TABLE/TR/TD/P/TA
BLE/TR
Document/html/BODY/TABLE/TR/TD/LI/FONT
Document/html/BODY/TABLE/TR/TD/TABLE/FORM/TR/T
D
/SELECT
Document/html/BODY/TABLE/TR/TD/P/TABLE/TR/TD/TA
BLE/TR /TD
Document/html/BODY/FORM/TABLE/TR/TD/TABLE/TR/T
D/FONT
Document/html/BODY/TABLE/TR/TD/A/CENTER
Document/html/BODY/TABLE/TR/TD/P/BR
Time required for templete extraction using Dice coefficient: 294 Average MDL Cost For MinHash with Dice : 118.62825482116476

Numbers of Clusters and Execution Time Required



Algorithm	Execution Time	Number Of Cluste
Text MDL	7751.0	18
MinHash with Jaccard Coff	937.0	11
MinHash with Dice Coff	352.0	10

Table VIII: Clustering Comparison



NumberofD..	TextMDLTime	TextMDLCost	MinHashJTi..	MinHashJC..	DiceTime	DiceCost
0	0.0	0.0	1.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	1.0	0.0
2	7.0	66.0	3.0	0.0	1.0	0.0
3	0.0	0.0	0.0	0.0	5.0	0.0
4	0.0	0.0	0.0	0.0	12.0	0.0
6	0.0	0.0	0.0	0.0	14.0	8.1657842...
7	0.0	0.0	0.0	0.0	3.0	0.0
8	0.0	0.0	0.0	0.0	8.0	14.019550...
9	0.0	0.0	0.0	0.0	66.0	54.489935...
10	0.0	0.0	0.0	0.0	120.0	38.451124...
11	0.0	0.0	0.0	0.0	18.0	50.824915...
12	0.0	0.0	0.0	0.0	5.0	0.0
13	0.0	0.0	0.0	0.0	234.0	80.593704...
14	0.0	0.0	0.0	0.0	10.0	0.0
15	0.0	0.0	0.0	0.0	106.0	97.117662...
18	0.0	0.0	0.0	0.0	361.0	101.95287...
19	5614.0	6249.2171...	578.0	110.77466...	399.0	118.62825...
25	951.0	3848.0193...	0.0	0.0	0.0	0.0
30	2091.0	5249.9465...	0.0	0.0	0.0	0.0
32	17755.0	3102.7052...	138.0	109.56822...	0.0	0.0
34	4038.0	6905.5729...	28.0	0.0	0.0	0.0
40	63893.0	7010.8627...	332.0	194.18933...	145.0	0.0
44	68392.0	5803.2574...	0.0	0.0	0.0	0.0
46	[illegible]	[illegible]	0.0	0.0	0.0	0.0

Table VIII shows the clustering result of web documents by using TEXTMDL method, by using MinHash Jaccard coefficient method, also MinHash Dice coefficient method.

7. CONCLUSION

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. We introduced a novel approach of the template detection from heterogeneous web documents.

We implement the MDL principle to manage the unknown number of clusters. Select good partition from all partitions to documents. Then we implement our MinHash technique to speed up the clustering process. Experimental results with set of web documents effectively cluster.

8. REFERENCES

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

[2] A.Z. Broder, M. Charikar, A.M. Frieze, and M. Mitzenmacher, *Min-Wise Independent Permutations J. Computer and System Sciences, vol. 60, no. 3, pp. 630-659, 2000.*

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

[4] 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.*

[5] 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.*

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

[7] I.S. Dhillon, S. Mallela, AND D.S. Modha, *InformationTheoretic CO-Clustering. PROC. ACM SIGKDD, 2003*

[8] D. Gibson, K. Punera, AND A. Tomkins, *The Volume And Evolution Of Web Page Templates PROC. 14TH INT?L CONF. WORLD WIDE WEB (WWW), 2005.*

[9] B. Long, Z. Zhang, AND P.S. Yu, *Co-Clustering By Block Value Decomposition PROC. ACM SIGKDD, 2005*

[10] F. Pan, X. Zhang, AND W. Wang, *CRD: Fast CO-Clustering On Large Data Sets Utilizing Sampling-Based Matrix Decomposition. PROC. ACM SIGMOD, 2008*

[11] Kim And Shim, *Text: Automatic Template Extraction From Heterogeneous Web Pages. 'IEEE Transactions On Knowledge And Data Engineering, VOL. 23, NO. 4, APRIL 2011*

[12] Hanady Abdul Salam, David B. Skillicorn, *Classification Using Streaming Random Forests. IEEE Computer Society, IEEE Transactions On Knowledge And Data Engineering, VOL. 23, NO. 1, JANUARY 2011.*