Research Journal of Applied Sciences 9 (5): 288-294, 2014
ISSN: 1815-932X

Performance Intensification for Automatic Template Using World Wide Web

[1]G. NaveenSundar, [1]D. Narmadha and [2]A.P. Haran
[1]School of Computer Science and Technology, Karunya University,
641114 Coimbatore, India
[2]Department of Mechanical and Aeronautical Engineering Park,
College of Technology and Science, Coimbatore, India

Abstract: Every individual is provided with access to plenty of information with the help of world wide web but it becomes progressively more difficult to discover the significant pieces of information. In web mining tries to tackle this problem by applying data mining techniques to web data and documents. The data available on the web is so heterogeneous and huge that it becomes a crucial factor to extract this accessible data to make it pertinent to a particular problem. Web mining uses data mining techniques to extract knowledge from web sources. This study focuses on detecting and extracting templates from web pages that are heterogeneous in nature by means of an algorithm. Locality sensitive hashing finds the similarity between the input web documents and provides good performance compared to the Minimum Description Length (MDL) principle and hash cluster process in terms of execution time.

Key words: Cluster, non-content path, template detection, Minimum Description Length (MDL), web

INTRODUCTION

The most widely used knowledge provider is the world wide web or www. Web pages on the websites are constructed in such a way that almost 50% of the data contains the 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 the look and feel of web pages. At the same time, the presences of templates in very large amount of 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 study 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 study, 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.

LITERATURE REVIEW

There are different methods available for template detection and extraction. Many of the earlier methods (Bar-Yossef and Rajagopalan, 2002; Vieira *et al*., 2006; Yi *et al*., 2003) were based on the assumption that all the web pages belong to a common template structure. The use of factors like Tree-edit distance (Vieira *et al*., 2006; Ma *et al*., 2003) 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 (Weninger *et al*., 2010). 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 (Weninger *et al*., 2010). This reduces the effectiveness of the old content extraction techniques.

Corresponding Author: G. NaveenSundar, School of Computer Science and Technology, Karunya University, 641114 Coimbatore, India

Many existing approaches (Gupta *et al*., 2003) uses filtering techniques such as advertisement remover, link list remover but it does not find content but eliminates non-content. Furthermore, in the method proposed (Bar-Yossef and Rajagopalan, 2002; Chen *et al*., 2006; Arasu and Garcia-Molina, 2003) use frequency of words as similarity measure. Template detection may be based on a threshold value (Ma *et al*., 2003) for the frequency of text in documents. Some of the earlier approaches require large human intervention for collecting training examples (Jushmerick, 1999) in order to distinguish between actual content and templates. A page-level (Wang *et al*., 2008) 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.

REPRESENTATIONS OF THE 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 study represent web documents in the form of matrices. As an example, consider two web documents represented using HTML tags.

Sample Web document: <html>, </html>, <body>, <html>, <h1>IT</h1>, <body>
, <h1>Park, </body>, </h1>,
, Gate, </body>.

The 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 1: Paths and pathno values

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

IDENTIFYING NON-CONTENT PATHS

The web documents are given a threshold value known as least path no threshold value. It is calculated as the mode of path no 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 path no threshold value specified in 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 where PDOC is the path set and DOC is the documentation set. A value of 1 at the ith row and jth 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.

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

A type of hierarchical clustering known as agglomerative hierarchical clustering is used in order to cluster similar documents together in various clusters or groups. The method repeatedly merges documents in clusters until one single cluster is formed. A cluster cl_i is represented as a pair ($TEMP_i$, DOC_i) where $TEMP_i$ is the template path set of the cluster and DOC_i is the member document sets of the cluster. A whole clustering model is represented as 2 matrices MtTEMP and MtDOC with dimensions |PDOC|*|DOC| and |DOC|*|DOC|, respectively. MtTEMP indicates template paths in the cluster formed and MtDOC indicates member documents present in the cluster. A value of 1 at the ith row and jth column of MtTEMP indicates that path, $path_i$ is a template path of cluster cl_j. Similarly, a value of 1 in the ith row and jth column of MtDOC indicates that document doc_j is a member of cluster cl_i.

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:

$$H(X) = \sum -P(x)\log_2 P(x) \quad x \in \{0, 1, -1\} \qquad (1)$$

$$MDv(M_t) = |M_t|.H(X) \quad (2)$$

Where:
H(X) = The entropy of a random variable X in the matrix
P(x) = 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:
$Mdv(Mt_{TEMP})$ = The MDLval of matrix Mt_{TEMP}
$Mdv(Mt_{DOC})$ = The MDLval of matrix Mt_{DOC}

MDL principle states that if 2 clustering models CL_1 and CL_2 are considered, the cluster with the lowest MDL value is taken as the best cluster. CL_1 is taken as the best cluster when compared to CL_2 if and only if $MDv(CL_1)$ is less than $MDv(CL_2)$.

Clustering 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 as:

Algorithm extract template (DOC):

```
begin
1. CL: = {cl1, cl2, ….cln}with cli = (NC(doci),{doci});
2. (cli, clj, clk): = FindBestCluster(CL);
3. While (cli, clj, clk) is not null do {
4. CL: = CL-{cli,clj}U{clk};
5. (cli, clj, clk): = FindBestCluster(CL);}
6. return CL
End

procedure FindBestCluster(CL)
begin
1. MDLvallowest: = ∞;
2. For each pair (cli, clj) of clusters in CL do {
3. (MDLval, clk): = CalcMDLval (cli, clj, CL);
4. If MDLval<MDLvallowest then {
5. MDLvallowest: = MDLval;
6. (cliB, cljB, clkB): = (cli, clj, clk);
7. }}
8. return (cliB, cljB, clkB);
End

procedure CalcMDLval (cli, clj, CL)
begin
1. DOCk: = DOCi U DOCj;
2. TEMPk: = {pathx|ndoc(pathx, DOCk)> = |DOCk|+1/2, pathxNCk};
3. clk: = (TEMPk,DOCk);
4. CL': = CL-{cli, clj}U{clk};
5. MDL: = MDL value of CL' ;
6. return (MDL, clk);
End
```

where, CL indicates the whole clustering model, cl_1, cl_2,... indicates individual clusters, NC (doc_i) represents non-content path of doc_i, cl_k 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 cl_i 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, $path_x$ is a non-content path.

Clustering using MinHash: A cluster is chosen as the best one if it's reduction of MDLcost 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 as:

```
procedure: GetInitBestPair (C)
begin
Merge all clusters with the same signature of MinHash;
MDLmin: = ∞;
For each ci in C do {
N: = clusters with the maximal Jaccard's coefficient with ci;
for each cj in N do {
 (MDLtmp,ck): = GetHashMDLcost (ci, cj, C);
 If MDLtmp<MDLmin then {
MDLmin: = MDLtmp,
(ciB, cjB, ckB): = (ci, cj, ck);
}}
return (ciB, cjB, ckB);
 end
procedure: GetHashBestPair (ck, C)
begin
(ciB, cjB ): = the current best pair;
ckB: = a cluster made by merging ciB and cjB ;
MDLmin: = the current best MDLcost;
N: = clusters with the maximal Jaccard's coefficient with ck;
for each ci in N do {
(MDLtmp,ctmp): = GetHashMDLcost (ck, ci, C);
If MDLtmp<MDLmin then {
MDLmin: = MDLtmp,
(ciB, cjB, ckB): = (ck, ci, ctmp);
}}
 return (ciB, cjB, ckB);
 end
procedure: GetHashMDLcost (ci, cj, C)
begin
Dk : = Di U Dj, ck: = (Ø, Dk), C' = {ci, cj} U {ck};
for each Πq in Π do {
r (sigDk[q] : = min (r(sigDi[q]), r(sigDj[q]));
if (r(sigDi[q]) = = r(sigDj[q]) then
n (sigDk[q] : = n (sigDi[q])+n (sigDj[q]);
else n (sigDk[q]) is from the less one}.
Calculate ε (Dk, l) ;
MDL : = MDLcost;
return (MDL, ck);
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.

The Enhanced algorithm when the hashing concept is included is LSH algorithm. The current algorithm is the same as the basic approach with slight modifications.

Algorithm (TEXT-MDL(D)):

begin
C: = {c_1, c_2, ..., c_n} with c_i = (E(d_i),{d_i});
(c_i, c_j, c_k): = GetBestPair(C);
While (c_i, c_j, c_k) is not empty do {
C: = C-{c_i, c_j}U{c_k};
(c_i, c_j, c_k): = GetBestPair(C);}
return C
end
procedure: GetBestPair(C)
begin
$MDLcost_{min}$: = ∞;
For each pair (c_i,c_j) of clusters in C do {
(MDLcost, c_k): = GetLSHMDLcost (c_i, c_j, C);
If MDLcost<$MDLcost_{min}$ then {
$MDLcost_{min}$: = MDLcost;
(c_i^B, c_j^B, c_k^B): = (c_i, c_j, c_k);
}}
return (c_i^B, c_j^B, c_k^B);
end

procedure: GetLSHMDLcost (c_i, c_j, C)
begin
D_k: = D_i U D_j, C' = C- {c_i, c_j} U c_k;
Compute hash function.
Compare hash values of documents.
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.
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.
Compute Pr(1) and Pr(-1) in M_T and M_D.
MDL: = MDLcost.
return (MDL, c_k);
end

procedure: GetBestPair (c_k,C)
begin
(c_i^B, c_j^B): = the current best pair;
c_k^B: = a cluster made by merging c_i^B and c_j^B;
MDL_{min}: = the current best MDLcost;
For each c in C do {
(MDL_{tmp},c_{tmp}): = GetLSHMDLcost (c_k,c_l,C);
If MDL_{tmp}<MDL_{min} then {
MDL_{min}: = MDL_{tmp};
(c_i^B, c_j^B, c_k^B): = (c_k, c_l, c_{tmp});
}}
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 a hash function. The computed hash values are then compared to the documents and thus similarity is found out.

BLOCK DIAGRAM OF TEMPLATE EXTRACTION

The block of the template extraction of the whole process is shown in Fig. 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 Fig. 1. Data sets are taken from five different websites 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 they 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 final the performances of all the methods are compared.

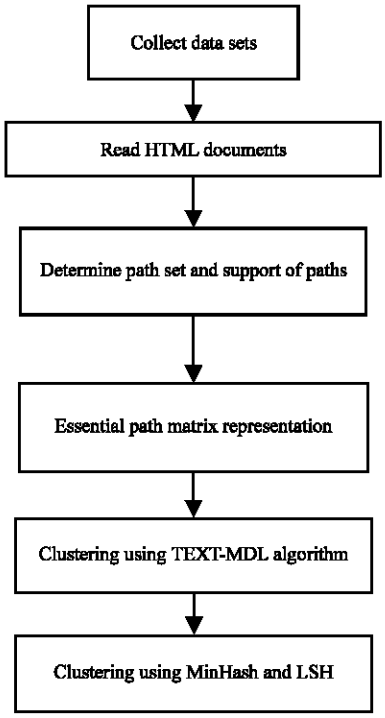

Fig. 1: Architecture of template extraction

EXPERIMENTAL ANALYSIS

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

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 an analysis, a set of five documents are taken and their results are analyzed as shown in Table 2 and the execution time taken is 5 sec for TEXT-MDL, 3 sec for TEXT-HASH and 1 sec for locality sensitive hashing for an input set of five documents. The memory required is 5232 bytes for TEXT-MDL, 3223 bytes for TEXT-HASH and 1242 bytes for locality sensitive hashing for the same set of input documents.

Comparison between TEXT-MDL and Text-Hash: The results are analyzed 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 Fig. 2.

Table 2: Result analysis

Methods	No. of documents	Execution time (sec)	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

In Fig. 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.

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

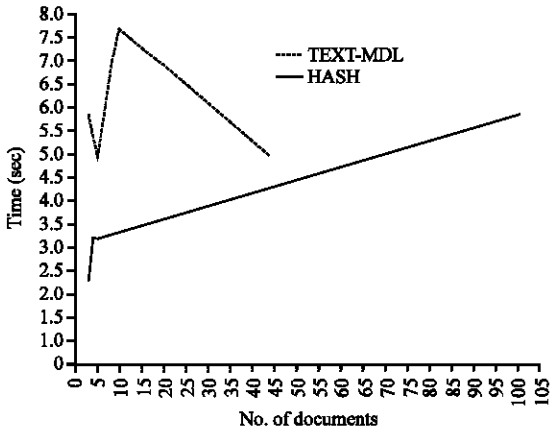

Fig. 2: Time comparison between MDL and HASH

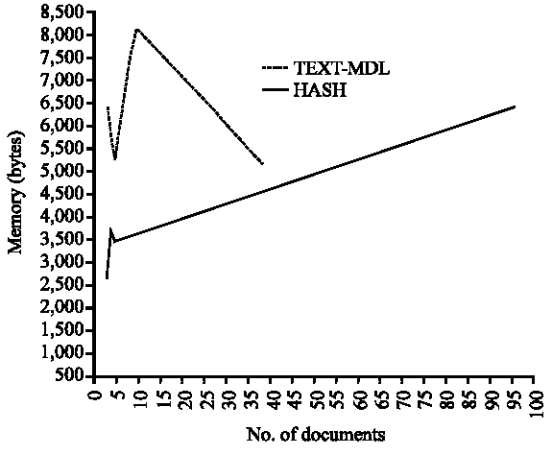

Fig. 3: Memory comparison between MDL and HASH

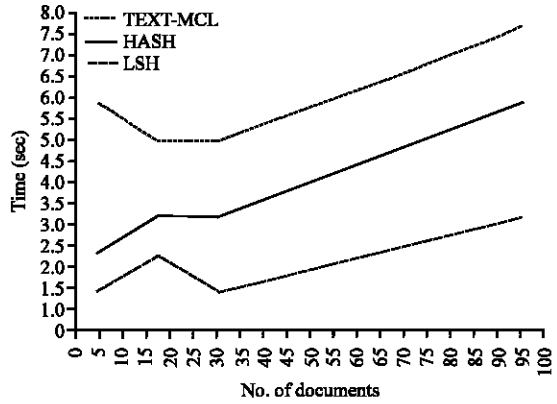

Fig. 4: Time comparison between MDL, HASH and LSH

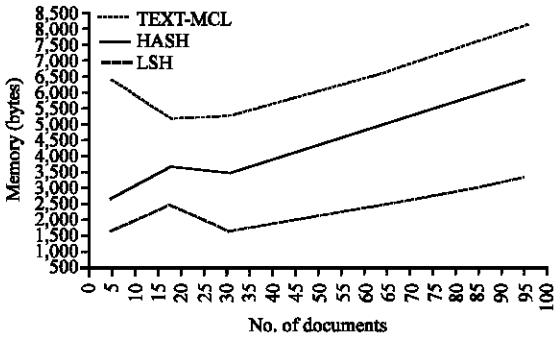

Fig. 5: Memory comparison between MDL, HASH and LSH

calculate the similarity between documents. Hence, the performance of TEXT-HASH is very much better when compared to TEXT-MDL.

Comparison between Text-MDL, Text-HASH and LSH: The time comparison between all the three approaches is shown in Fig. 3. The results are analyzed for various numbers of input documents and based on the values obtained, performance is compared.

A drastic reduction in execution time 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 (Fig. 4).

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

The memory comparison graph is shown in Fig. 5. 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.

CONCLUSION

The current research involves extracting templates from web pages automatically. The implementation of the research 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 research 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 research. Therefore, as a future research, 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

Arasu, A. and H. Garcia-Molina, 2003. Extracting structured data from web pages. Proceedings of the ACM SIGMOD International Conference on Management of Data, June 9-12, 2003, San Diego, CA., pp: 337-348.

Bar-Yossef, Z. and S. Rajagopalan, 2002. Template detection via data mining and its applications. Proceedings of the 11th International Conference on World Wide, May 7-11, 2002, Honolulu, Hawaii, USA., pp: 580-591.

Chen, L., S. Ye and X. Li, 2006. Template detection for large scale search engines. Proceedings of the ACM Symposium on Applied Computing, April 23-27, 2006, Dijon, France, pp: 1094-1098.

Gupta, S., G. Kaiser, D. Neistadt and P. Grimm, 2003. DOM based content extraction of HTML documents. Proceedings of the 12th international conference on World Wide Web, May 20-24, 2003, NewYork, USA., pp: 207-214.

Jushmerick, N., 1999. Learning to remove internet advertisements. Proceedings of the 3rd Annual Conference on Autonomous Agents, May 1-5, 1999, Seattle, WA., USA., pp: 175-181.

Ma, L., N. Goharian, A. Chowdhury and M. Chung, 2003. Extracting unstructured data from template generated web documents. Proceedings of the 12th International Conference on Information and K nowledge Management, November 3-8, 2003, New Orleans, Louisiana, USA., pp: 512-515.

Vieira, K., A.S. da Silva, N. Pinto, E.S. de Moura, J.M.B. Cavalcanti and J. Friere, 2006. A fast and robust method for web page template detection and removal. Proceedings of the 15th ACM International Conference on Information and Knowledge Management, November 5-11, 2006, Arlington, Virginia, USA., pp: 258-267.

Wang, Y., B. Fang, X. Cheng, G. Li and H. Xu, 2008. Incremental web page template detection by text segments. Proceedings of the IEEE International Workshop on Semantic Computing and Systems, July 14-15, 2008, Huangshan, pp: 174-180.

Weninger, T., W.H. Hsu and J. Han, 2010. CETR: Content extraction via tag ratios. Proceedings of the 19th International Conference on World Wide Web, April 26-30, 2010, Raleigh, North Carolina, USA., pp: 971-980.

Yi, L., B. Liu and X. Li, 2003. Eliminating noisy information in web pages for data mining. Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 24-27, Washington, DC. New York, pp: 296-305.