Web Document Clustering Using Document Index Graph

B. F. Momin [1] *Student Member IEEE*, P. J. Kulkarni [2], Amol Chaudhari [3]
[1]bfmomin@rediffmail.com, [2] pjk_walchand@rediffmail.com, [3] amolac_02@yahoo.com
Dept. Of Computer Science & Engineering,
Walchand College Of Engineering, SANGLI, INDIA.

Abstract

Document Clustering is an important tool for many Information Retrieval (IR) tasks. The huge increase in amount of information present on web poses new challenges in clustering regarding to underlying data model and nature of clustering algorithm. Document clustering techniques mostly rely on single term analysis of document data set. To achieve more accurate document clustering, more informative feature such as phrases are important in this scenario. Hence first part of the paper presents phrase-based model, Document Index Graph (DIG), which allows incremental phrase-based encoding of documents and efficient phrase matching. It emphasizes on effectiveness of phrase-based similarity measure over traditional single term based similarities. In the second part, a Document Index Graph based Clustering (DIGBC) algorithm is proposed to enhance the DIG model for incremental and soft clustering. This algorithm incrementally clusters documents based on proposed cluster-document similarity measure. It allows assignment of a document to more than one cluster. The DIGBC algorithm is more efficient as compared to existing clustering algorithms such as single pass, K-NN and Hierarchical Agglomerative Clustering (HAC) algorithm.

1. INTRODUCTION

The World Wide Web is rapidly emerging as an important medium for the dissemination of information related to wide range of topics. This increases need of techniques to unveil inherent structure in the underlined data. Clustering is one of these. Clustering enables one to discover hidden similarity and key concepts. Any clustering technique relies on concepts such as a data representation model, a similarity measure, a cluster model, a clustering algorithm.

Most of the document clustering techniques are based on single term based models such as vector space model [2]. These methods use similarity measures such as cosine measure [2] or jaccard measure [2]. These methods make use of single term analysis only and do not use word-proximity feature or phrase-based analysis.

The work is part of the project titled "Data Mining for Very Large Databases (VLDB)" funded under Research Promotion Scheme [RPS], All India Council for Technical Education [AICTE], New Delhi, INDIA. The author B. F. Momin is Principal Investigator of this project.

Though researchers tried to take advantage of phrase based model using different techniques such as Inductive Logic Programming (ILP)[3], by applying different NLP techniques, the results were not encouraging. This is because extraction of such phrases is computationally intensive task. Hence researchers focused on statistical phrase extraction [4]. Statistical phrase is represented by any sequence of words that appear continuously in text. N-grams (sliding window algorithm) [5] and suffix tree model [6] are used to extract statistical phrases. N-gram method suffers from drawbacks that it only considers fixed length phrases and as document size grows, dimensionality increases tremendously [6]. Suffix tree model finds out any length common phrases but it suffers from high redundancies stored in the form of suffixes.

To overcome these disadvantages K. Hammouda and S. Kamel proposed DIG model to find out matching phrases [1].

In this paper, we first used DIG model to demonstrate effectiveness of phrase based similarity over term based similarity, and then we proposed DIGBC algorithm to cluster documents efficiently. The paper is organized as follows: Section 2 reveals the structure assigned by HTML and weights assigned to different text parts accordingly. Section 3 introduces DIG model proposed by Hammouda et.al. [3]. Section 4 proposes new clustering algorithm (DIGBC) and section 5 contain set of experiments to demonstrate effectiveness of phrase-based similarity and efficiency of DIGBC algorithm. Section 6 concludes our work and mentions scope for future research.

2. WEB DOCUMENT STRUCTURE ANALYSIS

Most of the textual information on Web is represented using HTML. Structure imposed by HTML language identifies key parts of documents hence HTML tags can be used to assign different levels of importance to text appearing in different regions of the document [1]. Using this concept, HTML document can be divided in different regions, having different significance levels. To indicate these significance levels, we assigned text with some weight and encoded text information in XML format. In this system, we used three levels of significance viz: HIGH, MEDIUM and LOW. Text appearing in title, meta-data, and keywords has been assigned

HIGH, text appearing in bold, italics, section headings, table or image caption etc has been assigned MEDIUM and rest of text is considered as LOW. After restructuring of HTML document, document cleaning step is performed. This detects sentence boundaries, removes stopwords and performs stemming using Porter's stemming algorithm [8].

3. Document Index Graph

To achieve better clustering results, the underlying data model used, must capture salient features of the data. The proposed Document Index Graph (DIG) model indexes documents while maintaining sentence structure in the original document [1]. DIG also captures the significance level of original sentence and hence allows us to take advantage of structure imposed by HTML language as discussed in above section.

A. *Document Structure Overview*

The DIG is a directed graph G = (V, E) where,

V: is a set of nodes $\{v_1, v_2, \ldots, v_n\}$, where each node v represents a unique word in the entire document set; and

E: is set of edges $\{e_1, e_2, \ldots, e_n\}$, such that each edge e is an ordered pair of nodes (v_i, v_j) [1]. Edge (v_i, v_j) is from v_i to v_j, and v_j is adjacent to v_i.

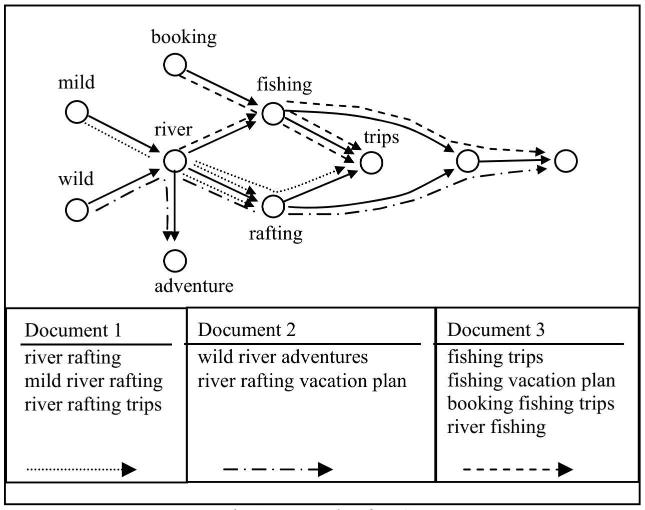
The above definition of the graph suggests that the number of nodes in the graph is the number of unique words in the document set. Each node contain sentence information such as: Document ID, Sentence Number, Word Number and Significance Level. Sentence structure is maintained by recording the edge and its positions in the corresponding sentences.

To better illustrate the graph structure, Fig. 1 presents a simple example graph that represents three documents. As seen from the graph, an edge is created between two nodes only if the words represented by the two nodes appear successive in any document. Thus, sentences map into paths in the graph. If a phrase appears more than once in a document, the sentence information in the node reflects the multiple occurrence of such phrase. As mentioned earlier, matching phrases between documents becomes a task of finding shared paths in the graph between different documents. This facilitates complete phrase matching of any-length [1].

B. *Similarity Measure*

Similarity between two documents is based on ratio of how much they overlap to their union in terms of phrases. Phrase based similarity measure proposed in [1] rewards phrases with high frequency, high significance level and greater length. Phrase based similarity measure is defined as below:

$$sim_p(d_1,d_2) = \frac{\sqrt{\sum_{i=1}^{P}\left[g(l_i)*(f_{1i}*w_{1i}+f_{2i}*w_{2i})\right]^2}}{\sum_j |s_{1j}|*w_{1j}+\sum_k |s_{2k}|*w_{2k}} \quad ......(1)$$



Document 1	Document 2	Document 3
river rafting mild river rafting river rafting trips	wild river adventures river rafting vacation plan	fishing trips fishing vacation plan booking fishing trips river fishing
⋯⋯▶	—·—·—▶	- - - - - -▶

Fig. 1: Example of DIG

Where,

P ← number of matching phrases,

L_i ← length of matching phrase,

f_{1i} & f_{2i} ← frequencies of phrase in both documents

w_{1i} * w_{2i} ← weight of phrase in both documents,

$|s_{1i}|$ ← length of ith sentence in document 1

$g(l_i)$ ← function which give higher score for phrases having length near to original sentence length. It is computed as

$$g(l_i) = l_i / |s_i| \quad ......(2)$$

But, sometimes similarity based on phrases only is not sufficient. To alleviate this problem, and to produce high quality clusters, it combines single-term similarity measure with phrase-based similarity measure. It uses the cosine correlation similarity measure [2], with TF-IDF (Term Frequency–Inverse Document Frequency) term weights [2], as the single-term similarity measure.

$$sim_t(d_1,d_2) = \frac{\sum_{t_i\in d_1 \& t_i\in d_2} TW(t_i,d_1)*TW(t_i,d_2)}{\sqrt{\sum_j TW(t_j,d_1)^2}*\sqrt{\sum_k TW(t_k,d_2)^2}} \quad ......(3)$$

Where, $TW(t_i,d_j)$ ← TF-IDF weight of ith term in jth document. The combination of phrase-based (eq. 1) and single-term (eq. 3) similarity is weighted average of the two quantities and is given as below:

$$sim(d_1,d_2) = \alpha * sim_p(d_1,d_2) + (1-\alpha) * sim_t(d_1,d_2) \quad ......(4)$$

Where, α is *similarity blend factor*. This determines contribution of phrase based similarity to total similarity. Similarity matrix generated using eq. 4 is further used to cluster documents using various clustering algorithms such as K-NN, HAC, single pass etc.

4. Document Index Graph Based Clustering (DIGBC)

Most of the document clustering techniques use the clustering algorithms such as K-NN [2], Hierarchical Agglomerative clustering [2], single pass [1] etc. These algorithms need to compute similarity of each document to every other document except in HAC which computes cluster

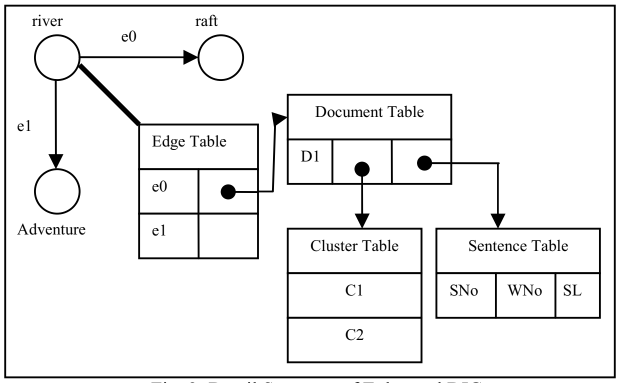


Fig. 2: Detail Structure of Enhanced DIG

similarity using centroid-based approach. But HAC also suffers from poor performance for large data set. Time complexity of all these algorithms is at least O(n2) [1][2]. As well as all these algorithms suffer from another drawback which shows that on introduction of new objects, whole object set must be re-clustered again. Also these algorithms assign each object to one fixed cluster hence they failed to detect multiple themes of a document. To overcome these drawbacks, we propose an incremental soft clustering algorithm: DIGBC. It allows addition of new documents without disturbing clusters already formed. It also allows a document to be added to multiple clusters. This algorithm is an extension to the DIG model discussed in section 3.

A. *Enhanced DIG*

In enhanced DIG, we encode cluster information related to the document with sentence information. The detail structure of enhanced DIG is shown in fig. 2. Enhanced DIG captures sentence information using edge table and sentence table. It also represents information regarding clusters to which each document belongs using cluster table. This additional information enables us to find out common phrases between current document and all existing clusters so that we can compute cluster-document similarity.

B. *Construction of Enhanced DIG and Phrase Matching*

Enhanced DIG is built incrementally i.e. by processing one document at a time. Algorithm 1 describes process of incremental DIG construction as well as phrase matching. When new document occurs, we process that document sequentially i.e. word by word. Add new words and edges to DIG. When an edge is searched or inserted in DIG, sentence table of that edge is modified to encode current sentence information. Simultaneously, we retrieve document table entries for that edge. Using these document table entries, we extend phrases from previous iteration if document id and sentence number are same and word number of current edge and phrases from previous iteration differ by one. Matching phrases are stored in list M which records length, number of documents in cluster which contain the phrase, frequency and significance level in cluster and current document. We add matching phrase to clusters to which documents sharing the phrase belongs. This matching phrase information is used further to compute cluster-document similarity.

```
Required: $G_{i-1} \leftarrow$ DIG up to document $d_{i-1}$
Or $G_0$ if no documents were processed previously.
1. $d_i \leftarrow$ next document
2. M $\leftarrow$ Empty List {list of matching phrase}
3. for each sentence $s_{ij}$ in $d_i$ do
4.    $v_1 \leftarrow t_{ij1}$ {first word in $s_{ij}$}
5.    if ($v_1$ is not in $G_{i-1}$) then
6.      add $v_1$ to $G_{i-1}$
7.    end if
8.    for each term $t_{ijk} \in s_{ij}$, k = 2, …, $l_{ij}$
9.      $v_k \leftarrow t_{ijk}$, $v_{k-1} \leftarrow t_{ij(k-1)}$, $e_k \leftarrow (v_{k-1}, v_k)$
10.     if ($v_k$ is not in $G_{i-1}$) then
11.       add $v_k$ to $G_{i-1}$
12.     else
13.       if $e_k$ is an edge in $G_{i-1}$
14.         Retrieve the list of document table entries from
15.          edge $e_k$'s table.
16.         Extend previous matching phrases in M for
17.           phrases that continue along edge $e_k$.
18.         Add new matching phrases to M.
19.       else
20.         add edge $e_k$ to $G_{i-1}$
21.       end if
22.     end if
23.     update sentence path in $v_{k-1}$ and $v_k$
24.   end for
25. end for
26. $G_i \leftarrow G_{i-1}$
27. Output matching phrase list M
```

Algorithm 1: Construction of Enhanced DIG & Phrase Matching Algorithm

C. *Complexity of Enhanced DIG*

As per the detailed structure of enhanced DIG, the number of graph nodes will be exactly same as number of distinct words in whole document set (say m). In worst case, number of edges in DIG becomes m^2. However, typically, the number of edges is around one order of magnitude larger than the number of nodes. Memory requirement of DIG is based on following factors:

n: is the number of documents in the data set,
m: is the number of unique terms in the data set,
$outdegree_{avg}$: is the average out-degree of a node, and
q: is the average number of terms per document,
idf_{avg} : average inverse document frequency of word
k : maximum number of clusters.

Then the space requirement of the model is

$$size(G) = (m * out\mathrm{deg}ree_{avg}) + 4 * q * n + idf_{avg} * k \quad ...(5)$$

First term in eq. 5 is space required for model without phrase indexing. Second term represents space required to encode phrase information and last term represents space required for encoding clustering information. As number of documents processed increases, number of new words and edges become very less. Maximum number of clusters (k) is also very small as compared to total number of documents. Hence space is required only to encode new sentence information. This makes size of DIG stable. Time complexity of construction of DIG mostly depends on time required for searching and inserting nodes and edges in the graph. But use of data structures such as hash table allows us to do this in near constant time. Hence time complexity of construction of DIG is directly proportional to total number of documents (n) and average words per document (q).

$$T(G) = O(q * n) \qquad ......(6)$$

Similarly time complexity of phrase matching algorithm depends on total number of documents (n) and average number of documents that share phrase with current document (idf_{avg}).i.e O(n*idf_{avg}). But, as it computes similarity among the document and existing clusters, time required to build similarity matrix and for clustering of document is reduced to O(n*nc). Where, 'nc' is maximum number of clusters present. Here in worst case, nc = n i.e. each document is assigned to separate cluster. But usually nc << n hence, this algorithm shows better performance.

D. *Similarity Measure*

The algorithm mentioned above finds out matching phrases between current document and all existing clusters. Here, we propose a similarity measure, which uses these matching phrases to find out cluster-document similarity.

$$sim(C,d) = \frac{\sqrt{\left(\sum_{i=1}^{p} g(l_i) * \left\{\frac{doc_i}{C.ndoc} * f_{ci} * w_{ci} + f_{di} * w_{di}\right\}\right)^2}}{CWT + DWT} \qquad ......(7)$$

Where,
C ← cluster with which similarity to be found out
d ← current document
p ← number of matching phrases
l_i ← length of ith phrase
doc_i ← number of documents in cluster C containing ith phrase
C.ndoc ← number of documents in cluster C
f_{ci}, w_{ci}, f_{di}, w_{di} ← frequency and weight of phrase in cluster C and document d.
CWT ← cluster normalization weight.
DWT ← document normalization weight.

$$CWT = \sum_{j \in C} DWT(d_j)$$

$$DWT(d_i) = \sum_{j \in C} | si_j | * w_{ij}$$

$g(l_i) = l_i / |s_i|$
$|s_i|$ ← length of original sentence.

E. *Creating Clusters with DIGBC*

In above sections, we described construction of enhanced DIG and phrase matching algorithm as well as we proposed cluster-documents similarity measure (eq. 7). Here we use this cluster-document similarity measure to produce clusters. The algorithm 2 describes how it performs clustering. As per this algorithm, whenever a new document arrives it first finds common phrases between the document and all existing clusters (line 2). For this, it uses matching phrase algorithm mentioned in previous section. The DIGBC algorithm, instead of storing matching phrases document wise, stores matching phrases cluster wise. Once the matching phrases are figured out, it calculates document-cluster similarity in line 3.

```
1.  di ← new document
2.  find out common phrases between di all existing clusters
3.  compute similarity of document to all existing clusters.
4.  for j = 0 to nclusters // nclusters ← number of existing
                            // clusters (initially 0)
6.      if (sim[di, cj] > threshold) then
8.          add di to cj
9.          modify document tables accordingly
10.     end if
11. end for
12. if di is not assigned to any existing cluster
13.     add di to a new cluster
15.     increment nclusters
16.     modify document tables accordingly.
17. end if
```

Algorithm 2: DIGBC Algorithm

The similarity is calculated using eq. 7. Then, it adds current document to every cluster which have similarity to document greater than certain threshold as mentioned in lines 4 to 11. If document can not be assigned to any of the existing clusters then it forms a new cluster and adds document to it (in line 12 to 17). When the document is assigned to any cluster, it modifies its document table accordingly (in lines 9& 16).

5. PERFORMANCE ANALYSIS

The Web Document Clustering System is implemented as an integration of all modules described in previous sections. We carried out some Experiment, to find out effectiveness of each module. The experiments conducted were divided into two sets. First test is performed to see the effectiveness of the DIG model, presented in section 3, and the accompanying phrase matching algorithm for calculating the similarity among documents based on phrases versus individual words only. The second set of experiments is carried out to evaluate the accuracy of the incremental document clustering algorithms, presented in section 4.

A. *Experimental Setup*

Here, we implemented this system using J2SDK v1.5 and all the experiments are carried out on Pentium IV (2.4GHz) machine. This system is tested on two data sets as described in table 1.

B. *Cluster Quality Evaluation Measure*

For clustering, two measures of cluster "goodness" or quality are used.

a) *F-measure*

The first external quality measure is the F measure, a measure that combines the precision and recall ideas from information retrieval [2].

TABLE 1: DATA SETS

Data Set	Name	# of Documents	Categories	Avg. words per doc
DS1	UW-CAN	314	10	469
DS2	Yahoo! News	2340	20	289

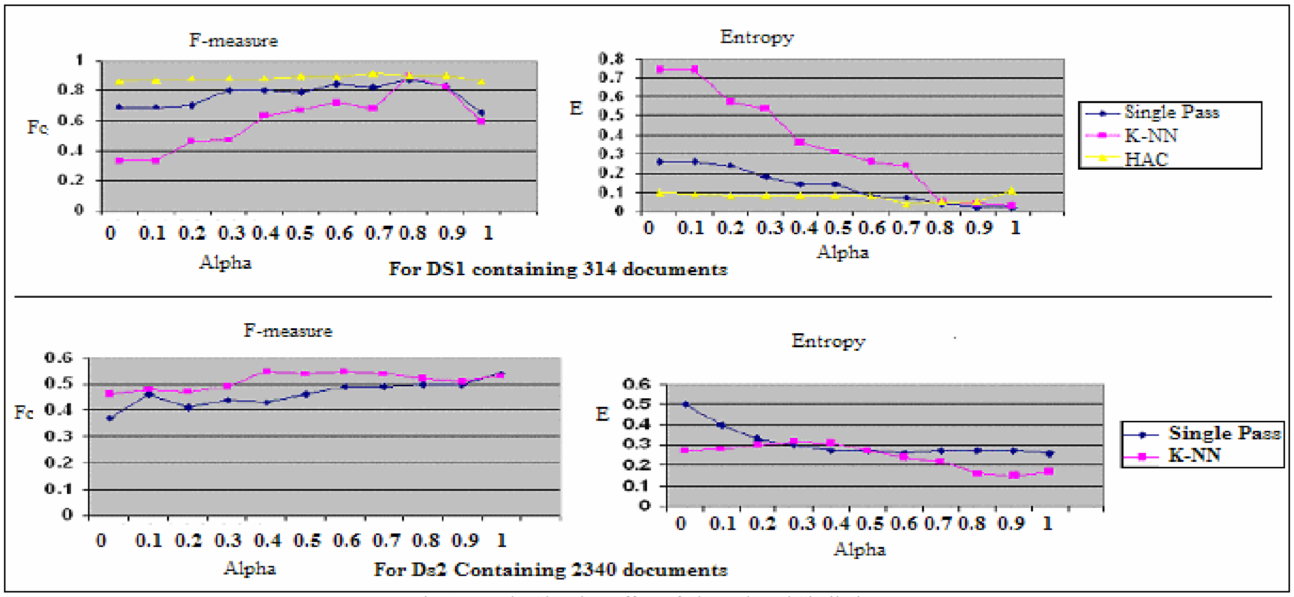


Fig. 3: Results Showing Effect of Phrase-based Similarity

Here, each cluster is treated as if it were the result of a query and each class is treated as if it were the desired set of documents for a query. Then, the recall and precision of that cluster for each given class is computed.

More specifically, for cluster *j* and class *i*

$$\text{Recall}(i, j) = n_{ij} / n_i$$

$$\text{Precision}(i, j) = n_{ij} / n_j \qquad ......(8)$$

Where,

n_{ij} is the number of members of class *i* in cluster *j*,

n_j is the number of members of cluster *j* and

n_i is the number of members of class *i*.

The F measure of cluster *j* and class *i* is then given by

F(i,j)=(2*P(i, j)*R(i, j))/((P(i,j)+R(i,j))

Overall value for the F measure is computed by taking the weighted average of all values for the F measure as given by the following.

$$F_c = \sum_i \frac{n_i}{n} * \max\{F(i,j)\} \qquad ......(9)$$

n ← total number of documents.

Higher the value of F-measure better is the cluster quality.

b) Entropy

It is another measure of quality of the clusters [2]. Let *CS* be a clustering solution. For each cluster, the class distribution of the data is calculated i.e., for cluster *j*, p_{ij}, the "probability" that a member of cluster *j* belongs to class *i*, is computed. Then using this class distribution, the entropy of each cluster *j* is calculated using the standard formula

$$E_j = -\sum_i p_{ij} * \log(p_{ij}) \qquad ......(10)$$

The total entropy for a set of clusters is calculated as

$$E_{cs} = \sum_{j=1}^{m} \frac{n_i * E_j}{n} \qquad ......(11)$$

Where,

n_j ← size of cluster j,

n ← total number of documents,

m ← total number of clusters.

Lower the entropy represents higher homogeneity of cluster. Hence we try to minimize entropy value for better cluster quality.

C. *Effect of Phrase-based Similarity on Clustering Quality*

The similarities calculated by eq. 4, are used to construct a similarity matrix between the documents. Three standard document clustering techniques are used for testing the effect of phrase-based similarity on clustering:

1. Hierarchical Agglomerative Clustering (HAC),
2. Single Pass Clustering, and
3. K-Nearest Neighbor Clustering (K-NN).

In order to understand the effect of the phrase similarity on the clustering quality, a clustering quality profile against the similarity blend factor is generated.

Fig. 3 illustrates the effect of introducing the phrase similarity on the F-measure and the entropy of the resulting clusters. It is obvious that the phrase similarity enhances the quality of clustering until a certain point (around a weight of 0.8) and then its effect starts bringing down the quality. Phrases alone cannot capture all the similarity information between documents, the single-term similarity is still required, but to a smaller degree.

The results show the improvement in the clustering quality on the first data set using the combined similarity measure. The improvements shown were achieved at a similarity blend factor between 0.7 and 0.8. The percentage of improvement ranges from 5.81 to 169 percent increase in the F-measure quality, and 60 to 95.94 percent drop in Entropy. It is obvious that the phrase based similarity plays an important role in accurately judging the relation between documents.

TABLE 2: RESULT FOR COMPARISON OF DIGBC

Algorithm	F-Measure	Entropy
	DS1	
DIGBC	0.87	0.16
Single Pass	0.67	0.02
K-NN	0.59	0.03
HAC	0.86	0.1
	DS2	
DIGBC	0.530	0.20
Single Pass	0.544	0.269
K-NN	0.531	0.170

The performance of the model was closely examined to make sure that the phrase matching algorithm is scalable enough for moderate to large data sets. Fig. 4 shows the performance of the graph construction and phrase matching algorithm for the larger data sets (DS2). The algorithm performed in a near-linear time.

D. Evaluation of DIGBC

The DIGBC method is applied on two data sets namely, DS1 and DS2. Then, it is evaluated using the F-measure and Entropy. From table 2, it is clear that DIGBC algorithm produces clusters having same quality as that from single pass and KNN algorithms. But DIGBC has greater value for entropy as compared to other algorithms that means it allows certain false clustering. For this experiment cluster-document threshold is 0.25. If the threshold value is adjusted crucially we may get good results for entropy. To evaluate performance of DIGBC algorithm, time required to build DIG should also be considered. Because in any other clustering algorithm, DIG is first created to compute similarity matrix for document set. From fig. 4 it is clear that this algorithm performs in near-linear time. Also if we compare time required for DIG creation from section 5.3 with this, it showed better performance. Hence this algorithm is scalable for moderate to large dataset.

6. CONCLUSION AND FUTURE RESEARCH

The system presented is extension of Document Index Graph Model. First part of system uses the DIG model to measure weighted phrase-based similarity between web documents. Such a model performs phrase matching and similarity calculation between documents in a very robust, efficient, and accurate way. The quality of clustering achieved using this model significantly surpasses the traditional vector space model based approaches. The second part is extension made to DIG model. This extension allows us to embed clustering algorithm in DIG construction and phrase matching algorithm. To find out cluster-document similarity, a similarity measure is devised which appropriately weight the factors affecting similarity value. This shows better performance in exchange of small extra storage space. Potential applications of this framework include automatic grouping of search engine results, phrase-based information retrieval, document similarity measurement, detection of plagiarism and many others. There are a number of future

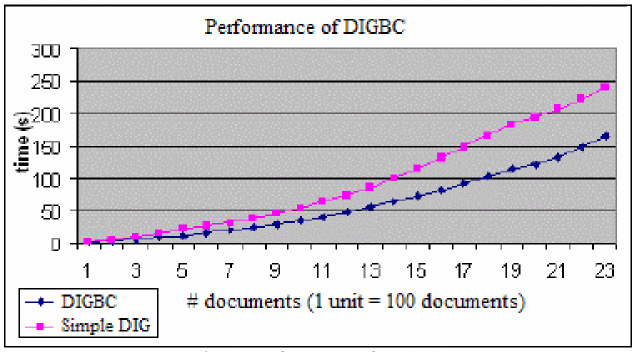


Fig. 4: Performance of DIGBC

research directions to extend and improve in this work. One direction that this work might continue on is to improve on the accuracy of document-cluster similarity calculation and the threshold value determination to achieve better quality. Although the work presented here is aimed at Web document clustering, it could be easily adapted to any document type as well.

REFERENCES

[1] Khaled M. Hammouda, Mohamed S. Kamel, "Efficient Phrase-Based Document Indexing For Web Document Clustering", IEEE transaction Oct 2004.

[2] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," Proc. KDD-2000 Workshop Text Mining, Aug. 2000. pp. 3-7.

[3] Mary Elaine Calif and Raymond J. Mooney, "Applying ILP-based Techniques to Natural Language Information Extraction: An Experiment in Relational Learning", Working notes of the IJCAI-97 workshop on Frontier of Inductive Logic Programming, pp. 7-11, Nagoya, Japan, 1997.

[4] M.F. Caropreso, S. Matwin, and F. Sebastiani, "Statistical Phrases in Automated Text Categorization," Technical Report IEI-B4-07-2000, Pisa, Italy, 2000, pp. 1-15.

[5] Min-Soo Kim, Kyu-Young Whang, Jae-Gil [6]. Lee, Min-Jae Lee, "n-Gram/2L: A Space and Time Efficient Two-Level n-Gram Inverted Index Structure", Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005, pp. 325-328.

[6] Sven Meyer zu Eissen, Benno Stein, Martin Potthast, "The Suffix Tree Document Model Revisited", Tochtermann, Maurer (Eds.): Proceedings of the I-KNOW '05, Graz, 5th International Conference on Knowledge Management, Journal of Universal Computer Science, pp. 596-603, ISSN 0948-695x.

[7] D. Lin, "An Information-Theoretic Definition of Similarity," Proc. 15th Int'l Conf. Machine Learning, pp. 296-304, 1998.

[8] M. F. Porter, "An Algorithm For Suffix Stripping", *Program*, vol. 14, no. 3, pp. 130-137, July 1980.