Phrase-based Document Similarity Based on an Index Graph Model

Khaled M. Hammouda Mohamed S. Kamel

Department of Systems Design Engineering
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
E-mail: {hammouda,mkamel}@pami.uwaterloo.ca

Abstract

Document clustering techniques mostly rely on single term analysis of the document data set, such as the Vector Space Model. To better capture the structure of documents, the underlying data model should be able to represent the phrases in the document as well as single terms. We present a novel data model, the Document Index Graph, which indexes web documents based on phrases, rather than single terms only. The semi-structured web documents help in identifying potential phrases that when matched with other documents indicate strong similarity between the documents. The Document Index Graph captures this information, and finding significant matching phrases between documents becomes easy and efficient with such model. The similarity between documents is based on both single term weights and matching phrases weights. The combined similarities are used with standard document clustering techniques to test their effect on the clustering quality. Experimental results show that our phrase-based similarity, combined with single-term similarity measures, enhances web document clustering quality significantly.

1. Introduction

In an effort to keep up with the tremendous growth of the World Wide Web, many research projects target the organization of such information in a way that will make it easier for the end users to find the information they want efficiently and accurately. Information on the web is present in the form of text documents (formatted in HTML), and that is the reason many web document processing systems are rooted in text data mining techniques.

Text data mining shares many concepts with traditional data mining methods. Data mining includes many techniques that can unveil inherent structure in the underlying data. One of these techniques is clustering. Applied to text data, clustering methods try to identify inherent groupings of the text documents so that a set of clusters are produced in which clusters exhibit high intra-cluster similarity and low inter-cluster similarity [2]. Generally speaking, text document clustering methods attempt to segregate the documents into groups where each group represents some topic that is different than those topics represented by the other groups [3]. By applying text mining in the web domain, the process becomes what is known as *web mining*. There are three types of web mining in general, according to Kosala *et al* [9]: (1) web structure mining; (2) web usage mining; and (3) web content mining. We are mainly interested in the last type, web content mining.

Any clustering technique relies on four concepts: (1) a data representation model, (2) a similarity measure, (3) a cluster model, and (4) a clustering algorithm that builds the clusters using the data model and the similarity measure. Most of the document clustering methods that are in use today are based on the Vector Space Model [1, 14, 13], which is a very widely used data model for text classification and clustering. The Vector Space Model represents documents as a feature vector of the terms (words) that appear in all the document set. Each feature vector contains term-weights (usually term-frequencies) of the terms appearing in that document. Similarity between documents is measured using one of several similarity measures that are based on such a feature vector. Examples include the *cosine* measure and the *Jaccard* measure. Clustering methods based on this model make use of single-term analysis only, they do not make use of any word proximity or phrase-based analysis[1]. The motivation behind the work in this paper is that we believe that document clustering should be based not only on single word analysis, but on phrases as well.

The work that has been reported in literature about using phrases in document clustering is limited. Most efforts have been targeted toward single-word analysis. The methods used for text clustering includes decision trees [11], statistical analysis [4], neural nets [5], inductive logic program-

[1] Throughout this paper the term "phrase" means a sequence of words, and not the grammatical structure of a sentence.

ming [8], and rule-based systems [15] among others. These methods are at the cross roads of more than one research area, such as database (DB), information retrieval (IR), and artificial intelligence (AI) including machine learning (ML) and Natural Language Processing (NLP).

The most relevant work to what is presented here is that of Oren Zamir *et al* [19, 20]. They proposed a phrase-based document clustering approach based on Suffix Tree Clustering (STC). The method basically involves the use of a "trie" (a compact tree) structure to represent shared suffixes between documents. Based on these shared suffixes they identify base clusters of documents, which are then combined into final clusters based on a connected-component graph algorithm. They claim to achieve $n \log(n)$ performance and produce high quality clusters. The results they showed were encouraging, but the suffix tree model could be argued to have a high number of redundancies in terms of the suffixes stored in the tree.

In this paper we propose a novel document representation model, the Document Index Graph (DIG) that captures the structure of sentences in the document set, rather than single words only. The DIG model is based on graph theory and utilizes graph properties to match any-length phrase from a document to any number of previously seen documents in a time nearly proportional to the number of words of the new document. This means that when a new document is introduced to the system, we can detect any-length phrase match from this document to all the previously seen documents in the data set by just scanning the new document and extracting the matching phrases from the document index graph.

The above model is used to measure the similarity between the documents using a new similarity measure that makes use of phrase-based matching. This similarity measure over-performs similarity measures that are based on single-word models only. The similarity calculation between documents is based on a combination of single-term similarity and phrase-based similarity. Similarity based on matching phrases between documents is proved to have a more significant effect on the clustering quality due to its insensitivity to noisy terms that could lead to incorrect similarity measure. Phrases are less sensitive to noise when it comes to calculating document similarity; this is due to the fact that it is less likely to find matching phrases in non-related documents (see section 4.1).

The clusters produced using this similarity combination were of higher quality than those produced using single-term similarity only. We relied on two clustering quality measures, the F-measure, and the Entropy of the clusters. Both of these quality measures were improved when phrase-similarity was introduced to the different clustering algorithms.

The rest of this paper is organized as follows. Section 2 discusses the important features of the semi-structured web documents. Section 3 introduces the Document Index Graph model. Section 4 presents the phrase-based similarity measure. Section 5 presents our experimental results. Finally we conclude and discuss future work in the last section.

2. Web document structure analysis

Web documents are known to be semi-structured. Since the HTML language is meant for specifying the layout of the document, it is used to present the document to the user in a friendly manner, rather than specify the structure of the data in the document, hence they are semi-structured. However, it is still possible to identify key parts of the document based on this structure. The idea is that some parts of the document are more informative than other parts, thus having different levels of significance based on where they appear in the document and the tags that surround them. It is less informing to treat the title of the document, for example, and the text body equally.

The proposed system analyzes the HTML document and restructures the document according into a predetermined structure that assigns different levels of significance to different document parts. The result is a well structured XML document that corresponds to the original HTML document, but with the significance levels assigned to the different parts of the original document.

Currently we assign one of three levels of significance to the different parts; HIGH, MEDIUM, and LOW. Examples of HIGH significance parts are the title, meta keywords, meta description, and section headings. Example of MEDIUM significance parts are text that appear in bold, italics, colored, hyper-linked text, image alternate text, and table captions. LOW significance parts are usually comprised of the document body text that was not assigned any of the other levels.

This structuring scheme is exploited in measuring the similarity between two documents (see section 4 for details). For example, if we have a phrase match of HIGH significance in both documents, the similarity is rewarded more than if the match was for LOW significance phrases. This is justified by arguing that a phrase match in titles, for example, is much more informative than a phrase match in body text.

A sentence boundary detector algorithm was developed to locate sentence boundaries in the documents. The algorithm is based on a finite state machine lexical analyzer with heuristic rules for finding the boundaries. Finally, a document cleaning step is performed to remove stop-words that have no significance, and to stem the words using the popular Porter Stemmer algorithm [12].

3. Document index graph

The vector space model does not represent any relation between the words, so sentences are broken down into their individual components without any representation of the sentence structure. On the other hand, the proposed Document Index Graph (DIG for short) indexes the documents while maintaining the sentence structure in the original documents. This allows us to make use of more informative phrase matching rather than individual words matching. Moreover, the DIG also captures the different levels of significance of the original sentences, thus allowing us to make use of sentence significance as well.

3.1. DIG structure

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

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 a set of *edges* $\{e_1, e_2, \ldots, e_m\}$, such that each edge e is an ordered pair of nodes (v_i, v_j). Edge (v_i, v_j) is from v_i to v_j , and v_j is adjacent to v_i. There will be an edge from v_i to v_j if, and only if, the word v_j appears successive to the word v_i in any document.

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; *i.e.* the vocabulary of the document set, since each node represents a single word in the whole document set.

Nodes in the graph carry information about the documents they appeared in, along with the sentence path information. Sentence structure is maintained by recording the edge along which each sentence continues. This essentially creates an inverted list of the documents, but with sentence information recorded in the inverted list.

Assume a sentence of m words appearing in one document consists of the following word sequence: $\{v_1, v_2, \ldots, v_m\}$. The sentence is represented in the graph by a path from v_1 to v_m, such that $(v_1, v_2)(v_2, v_3), \ldots, (v_{m-1}, v_m)$ are edges in the graph. Path information is stored in the vertices along the path to uniquely identify each sentence. Sentences that share sub-phrases will have shared parts of their paths in the graph that correspond to the shared sub-phrase.

The structure maintained in each node is a table of documents. Each document entry in the document table records the term frequency of the word in that document. Since words can appear in different parts of a document with different level of significance, the recorded term frequency is actually broken into those levels of significance, with a frequency count per level per document entry. This structure helps in achieving a more accurate similarity measure based on level of significance later on.

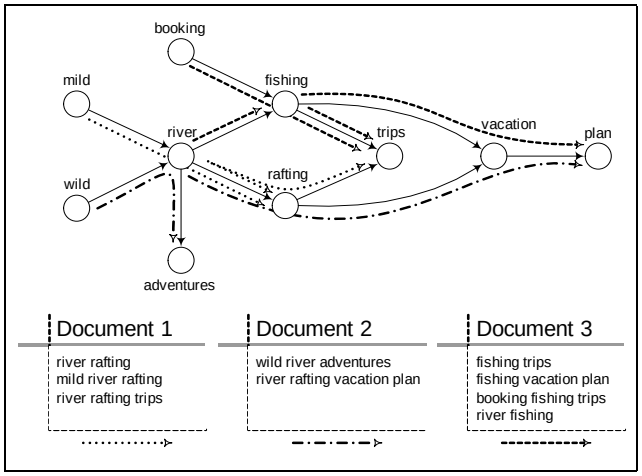


Figure 1. Example of the Document Index Graph

Since the graph is directed, each node maintains a list of an outgoing edges per document entry. This list of edges tells us which sentence continues along which edge. The task of creating a sentence path in the graph is thus reduced to recording the necessary information in this edge table to reflect the structure of the sentences.

To better illustrate the graph structure, Figure 1 presents a simple example graph that represents three documents. Each document contains a number of sentences with some overlap between the 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. Dotted lines represent sentences from document 1, dash-dotted lines represent sentences from document 2, and dashed lines represent sentences from document 3. If a phrase appears more than once in a document, the frequency of the individual words making up the phrase is increased, and the sentence information in the nodes 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.

The example presented here is a simple one. Real web documents will contain hundreds or thousands of words. With a very large document set, the graph could become more complex in terms of memory usage. Typically, the number of graph nodes will be exactly the same as the number of unique words in the data set. The number of edges is

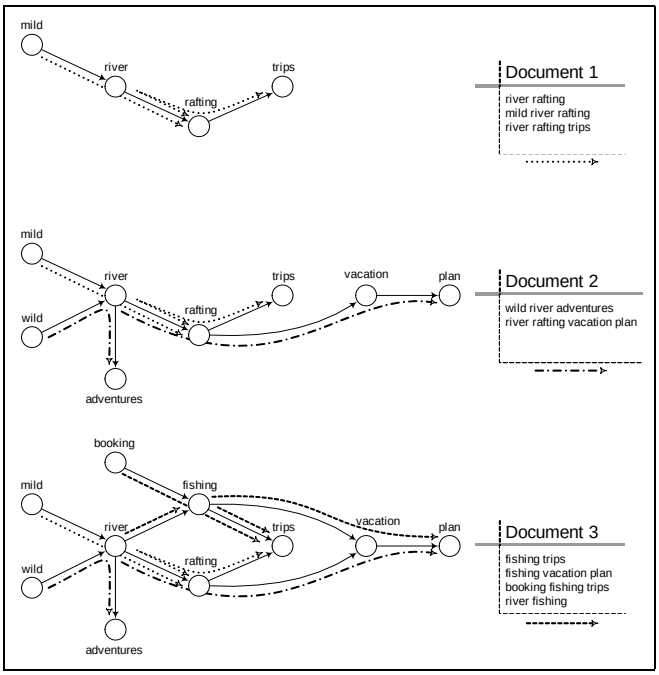


Figure 2. Incremental construction of the Document Index Graph

about 4 to 6 times the number of nodes (that is the average degree of a node).

3.2. Constructing the graph

The DIG is built incrementally by processing one document at a time. When a new document is introduced, it is scanned in sequential fashion, and the graph is updated with the new sentence information as necessary. New words are added to the graph as necessary and connected with other nodes to reflect the sentence structure. The graph building process becomes less memory demanding when no new words are introduced by a new document (or very few new words are introduced). At this point the graph becomes more stable, and the only operation needed is to update the sentence structure in the graph to accommodate for the new sentences introduced. It is very critical to note that introducing a new document will only require the inspection (or addition) of those words that appear in that document, and not every node in the graph. This is where the efficiency of the model comes from.

Along with indexing the sentence structure, the level of significance of each sentence is also recorded in the graph. This allows us to recall such information when we match sentences from other documents.

Continuing from the example introduced earlier, the process of constructing the graph that represents the three documents is illustrated in Figure 3.1. The emphasis here is on the incremental construction process, where new nodes are added and new edges are created incrementally upon introducing a new document.

Algorithm 1 Document Index Graph construction and phrase matching

1: $D \leftarrow$ New Document
2: **for** each sentence s in D **do**
3: $w_1 \leftarrow$ first word in s
4: **if** w_1 is not in G **then**
5: Add w_1 to G
6: **end if**
7: $L \leftarrow$ Empty List $\{L$ is a list of matching phrases$\}$
8: **for** each word $w_i \in \{w_2, w_3, \ldots, w_k\}$ in s **do**
9: **if** (w_{i-1}, w_i) is an edge in G **then**
10: Extend phrase matches in L for sentences that continue along (w_{i-1}, w_i)
11: Add new phrase matches to L
12: **else**
13: Add edge(w_{i-1}, w_i) to G
14: Update sentence path in nodes w_{i-1} and w_i
15: **end if**
16: **end for**
17: **end for**
18: Output matching phrases in L

Unlike traditional phrase matching techniques that are usually used in information retrieval literature, the DIG provides complete information about full phrase matching between *every* pair of documents. While traditional phrase matching methods are aimed at searching and retrieval of documents that have matching phrases to a specific query, the DIG is aimed at providing information about the degree of overlap between every pair of documents. This information will help in determining the degree of similarity between documents as will be explained in section 4.

3.3. Detecting matching phrases

Upon introducing a new document, finding matching phrases from previously seen documents becomes an easy task using the DIG. Algorithm 1 describes the process of both incremental graph building and phrase matching.

The procedure starts with a new document to process (line 1). We expect the new document to have well defined sentence boundaries; each sentence is processed individually. This is important because we do not want to match a phrase that spans two sentences (which could break the local context we are looking for.) It is also important to know the original sentence length so that it will be used in the similarity calculation (section 4). For each sentence (for loop at line 2) we process the words in the sentence sequentially,

adding new words as new nodes to the graph, and constructing a path in the graph (by adding new edges if necessary) to represent the sentence we are processing. At the beginning of each sentence we locate the first word of the sentence in the graph by consulting a hash table. If the word is in the graph, we continue from that node, otherwise we add it to the graph and link it to other nodes as necessary.

Matching the phrases from previous documents is done by keeping a list L that holds an entry for every previous document that shares a phrase with the current document D. As we continue along the sentence path, we update L by adding new matching phrases and their respective document identifiers, and extending phrase matches from the previous iteration (lines 10 and 11). If there are no matching phrases at some point, we just update the respective nodes of the graph to reflect the new sentence path (lines 13 and 14). After the whole document is processed L will contain all the matching phrases between the current document and any previous document that shared at least one phrase with the new document. Finally we output L as the list of documents with matching phrases and all the necessary information about the matching phrases.

The above algorithm is capable of matching *any-length* phrase between a new document D and all previously seen documents in roughly $O(m)$ time, where m is the number of words in document D. Some could argue that the step at line 10 in the algorithm, where we extend the matching phrases as we continue along an existing path, is not actually a constant time step, because when the graph starts building up, the number of matching phrases becomes larger, and consequently when moving along an existing path we have to match more phrases. However, it turns out that the size of the list of matching phrases becomes roughly constant even with very large document sets, due to the fact that a certain phrase will be shared by only a small set of documents; which on average tends to be a constant number.

4. A phrase-based similarity measure

As mentioned earlier, phrases convey local context information, which is essential in determining an accurate similarity between documents. Towards this end we devised a similarity measure based on matching phrases rather than individual terms. This measure exploits the information extracted from the previous phrase matching algorithm to better judge the similarity between the documents. This is related to the work of Isaacs *et al* [6] who used a pair-wise probabilistic document similarity measure based on Information Theory. Although they showed it could improve on traditional similarity measures, but it is still fundamentally based on the vector space model representation.

The phrase similarity between two documents is calculated based on the list of matching phrases between the two documents. This similarity measure is a function of four factors:

- The number of matching phrases P,
- The lengthes of the matching phrases ($l_i : i = 1, 2, \ldots, P$),
- The frequencies of the matching phrases in both documents (f_{i1} and $f_{i2} : i = 1, 2, \ldots, P$), and
- The levels of significance (*weight*) of the matching phrases in both document (w_{i1} and $w_{i2} : i = 1, 2, \ldots, P$).

Frequency of phrases is an important factor in the similarity measure. The more frequent the phrase appears in both documents, the more similar they tend to be. Similarly, the level of significance of the matching phrase in both documents should be taken into consideration.

The phrase similarity between two documents, $\mathbf{d_1}$ and $\mathbf{d_2}$, is calculated using the following empirical equation:

$$\mathrm{sim}_p(\mathbf{d_1}, \mathbf{d_2}) = \frac{\sqrt{\sum_{i=1}^{P}[g(l_i) \cdot (f_{i1}w_{i1} + f_{i2}w_{i2})]^2}}{\sum_j |s_{j1}| \cdot w_{j1} + \sum_k |s_{k2}| \cdot w_{k2}} \quad (1)$$

where $g(l_i)$ is a function that scores the matching phrase length, giving higher score as the matching phrase length approaches the length of the original sentence; $|s_{j1}|$ and $|s_{k2}|$ are the original sentence lengths from document $\mathbf{d_1}$ and $\mathbf{d_2}$, respectively. The equation rewards longer phrase matches with higher level of significance, and with higher frequency in both documents. The function $g(l_i)$ in the implemented system was used as: $g(l_i) = (|ms_i|/|s_i|)^\gamma$, where $|ms_i|$ is the matching phrase length, and γ is a sentence fragmentation factor with values greater than or equal to 1. If γ is 1, two halves of a sentence could be matched independently and would be treated as a whole sentence match. However, by increasing γ we can avoid this situation, and score whole sentence matches higher than fractions of sentences. A value of 1.2 for γ was found to produce best results.

The normalization by the length of the two documents in equation (1) is necessary to be able to compare the similarities from other documents.

4.1. Combining single-term and phrase similarities

If the similarity between documents is based solely on matching phrases, and not single-terms at the same time, related documents could be judged as non-similar if they do not share enough phrases (a typical case that could happen in many situations.) Shared phrases provide important local context matching, but sometimes similarity based

on phrases only is not sufficient. To alleviate this problem, and to produce high quality clusters, we combined single-term similarity measure with our phrase-based similarity measure. We used the *cosine correlation* similarity measure [14], with TF-IDF (Term Frequency–Inverse Document Frequency) term weights, as the single-term similarity measure. The cosine measure was chosen due to its wide use in the document clustering literature, and since it is described as being able to capture human categorization behavior well [17]. The TF-IDF weighting is also a widely used term weighting scheme [18].

Recall that the cosine measure calculates the cosine of the angle between the two document vectors. Accordingly our term-based similarity measure (sim_t) is given as:

$$\mathrm{sim}_t(\mathbf{d_1}, \mathbf{d_2}) = \cos(\mathbf{d_1}, \mathbf{d_2}) = \frac{\mathbf{d_1} \cdot \mathbf{d_2}}{\|\mathbf{d_1}\|\|\mathbf{d_2}\|} \quad (2)$$

where the vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ are represented as term weights calculated using TF-IDF weighting scheme.

The combination of the term-based and the phrase-based similarity measures is a weighted average of the two quantities from equations (1) and (2), and is given by equation (3).

$$\mathrm{sim}(\mathbf{d_1}, \mathbf{d_2}) = \alpha \cdot \mathrm{sim}_p(\mathbf{d_1}, \mathbf{d_2}) + (1 - \alpha) \cdot \mathrm{sim}_t(\mathbf{d_1}, \mathbf{d_2}) \quad (3)$$

where α is a value in the interval $[0, 1]$ which determines the weight of the phrase similarity measure, or, as we call it, the *Similarity Blend Factor*.

5. Experimental results

In order to test the effectiveness of phrase matching in determining an accurate measure of similarity between documents, we conducted a set of experiments using our proposed data model and similarity measure.

Our experimental setup consisted of 314 web documents collected from University of Waterloo various web sites, such as the Graduate Studies Office, Information Systems and Technology, Health Services, Career Services, Co-operative Education, and other Canadian web sites. The documents were classified according to their content into 10 different categories. In order to allow for independent testing and the reproduction of the results presented here, we decided to make the data available to others. The document collection can be downloaded at: `http://pami.uwaterloo.ca/~hammouda/webdata/`.

The similarities calculated by our algorithm were used to construct a similarity matrix between the documents. We elected to use three standard document clustering techniques for testing the effect of phrase similarity on clustering [7]: (1) Hierarchical Agglomerative Clustering (HAC), (2) Single Pass Clustering, and (3) *k*-Nearest Neighbor Clustering (*k*-NN)[2]. For each of the algorithms, we constructed the similarity matrix and let the algorithm cluster the documents based on the presented similarity matrix.

In order to evaluate the quality of the clustering, we adopted two quality measures widely used in the text mining literature for the purpose of document clustering [16]. The first is the **F-measure**, which combines the *Precision* and *Recall* ideas from the Information Retrieval literature. The precision and recall of a cluster j with respect to a class i are defined as:

$$P = \mathrm{Precision}(i, j) = \frac{N_{ij}}{N_i} \quad (4a)$$

$$R = \mathrm{Recall}(i, j) = \frac{N_{ij}}{N_j} \quad (4b)$$

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 class i is $F(i) = 2PR/(P + R)$. With respect to class i we consider the cluster with the highest F-measure to be the cluster j that maps to class i, and that F-measure becomes the score for class i. The overall F-measure for the clustering result C is the weighted average of the F-measure for each class i:

$$F_C = \frac{\sum_i (|i| \times F(i))}{\sum_i |i|} \quad (5)$$

where $|i|$ is the number of objects in class i. The higher the overall F-measure, the better the clustering, due to the higher accuracy of the clusters mapping to the original classes.

The second measure is the **Entropy**, which provides a measure of "goodness" for un-nested clusters or for the clusters at one level of a hierarchical clustering. Entropy tells us how homogeneous a cluster is. The higher the homogeneity of a cluster, the lower the entropy is, and vice versa. The entropy of a cluster containing only one object (perfect homogeneity) is zero.

For every cluster j in the clustering result C we compute p_{ij}, the probability that a member of cluster j belongs to class i. The entropy of each cluster j is calculated using the standard formula $E_j = -\sum_i p_{ij} \log(p_{ij})$, where the sum is taken over all classes. The total entropy for a set of clusters is calculated as the sum of entropies for each cluster weighted by the size of each cluster:

$$E_C = \sum_{j=1}^{m} (\frac{N_j}{N} \times E_j) \quad (6)$$

where N_j is the size of cluster j, and N is the total number of data objects.

[2] Although *k*-NNis mostly known to be used for classification, it has also been used for clustering (example could be found in [10]).

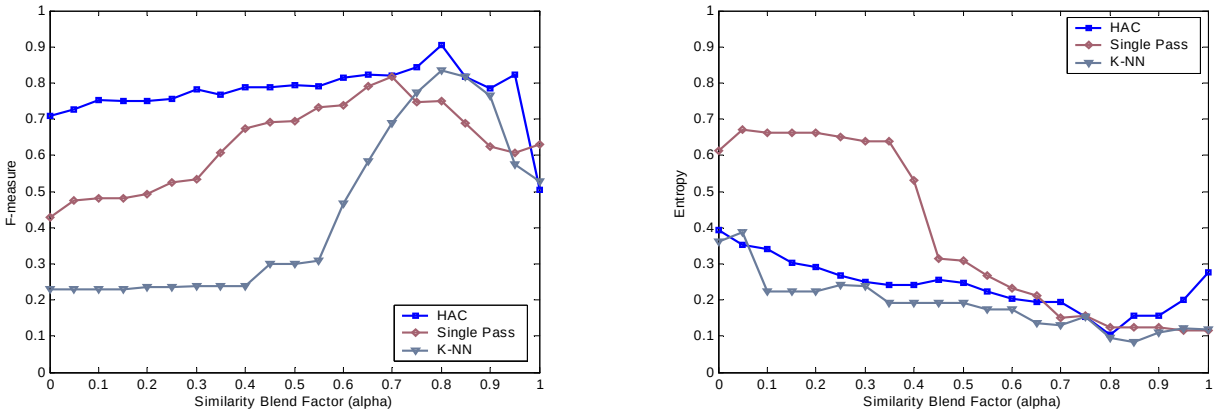
Table 1. Phrase-based clustering improvement

	Single-Term Similarity		Combined Similarity		Improvement
	F-measure	Entropy	F-measure	Entropy	
HAC [a]	0.709	0.351	0.904	0.103	+19.5%F, -24.8%E
Single Pass [b]	0.427	0.613	0.817	0.151	+39.0%F, -46.2%E
k-NN[c]	0.228	0.173	0.834	0.082	+60.6%F, -9.1%E

[a]Complete Linkage was used as the cluster distance measure for the HAC method since it tends to produce tight clusters with small diameter.

[b]A document-to-cluster similarity threshold of 0.25 was used.

[c]A k of 5 and a cluster similarity threshold of 0.25 were used.



(a) Effect of phrase similarity on F-measure

(b) Effect of phrase similarity on Entropy

Figure 3. Effect of phrase similarity on clustering quality

Basically we would like to maximize the F-measure, and minimize the Entropy of clusters to achieve high quality clustering.

The results listed in Table 1 show the improvement on the clustering quality using a combined similarity measure. The improvements shown were achieved at a similarity blend factor between 70% and 80% (phrase similarity weight—see section 4.1 for details on combining phrase-based similarity and single-term similarity.) The parameters chosen for the different algorithms were the ones that produced best results. The percentage of improvement ranges from 19.5% to 60.6% increase in the F-measure quality, and 9.1% to 46.2% drop in Entropy (lower is better for Entropy). It is obvious that the phrase based similarity plays an important role in accurately judging the relation between documents. It is known that Single Pass clustering is very sensitive to noise; that is why it has the worst performance. However, when the phrase similarity was introduced, the quality of clusters produced was pushed close to that produced by HAC and *k*-NN.

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

Figure 3(a) illustrates the effect of introducing the phrase similarity on the F-measure of the resulting clusters. It is obvious that the phrase similarity enhances the F-measure of the clustering until a certain point (around a weight of 80%) and then its effect starts bringing down the quality. As we mentioned in section 4.1 that phrases alone cannot capture all the similarity information between documents, the single-term similarity is still required, but to a smaller degree. The same can be seen from the Entropy profile in Figure 3(b), where Entropy is minimized at around 80% contribution of phrase similarity against 20% for the single-term similarity.

6. Conclusion and future research

We presented a system composed of three components in an attempt to improve the document clustering problem in the web domain. By exploiting the semi-structure inherent in web documents we can achieve better clustering results. We presented a web document analysis component that is capable of identifying the structure in web documents, and

building structured documents out of it.

The second component, and perhaps the most important one that has most of the impact on performance, is the new document model introduced in this paper, the Document Index Graph. This model is based on indexing web documents using phrases and their levels of significance. Such a model enables us to perform phrase matching and similarity calculation between documents in a very robust and accurate way. The quality of clustering achieved using this model significantly surpasses the traditional vector space model based approaches.

The third component is the phrase-based similarity measure. By carefully examining the factors affecting the degree of overlap between documents, we devised a phrase-based similarity measure that is capable of accurate calculation of pair-wise document similarity.

The merits of such a design is that each component could be utilized independent of the other. But we have confidence that the combination of these components leads to better results, as justified by the results presented in this paper. By adopting different standard clustering techniques to test against our model, we are very confident that this model is well justified, and not biased.

There are a number of future research directions to extend and improve this work. One direction that this work might continue on is to improve on the accuracy of similarity calculation between documents by employing different similarity calculation strategies.

Although the work presented here is aimed at web document clustering, it could be easily adapted to any document type as well. However, it will not benefit from the semi-structure found in web documents. Our intention is to investigate the usage of such model on standard corpora and see its effect on clustering compared to traditional methods.

An important part of the clustering process is the clustering algorithm itself. One important benefit of the presented model is that we can use it with any clustering algorithm that can make use of the accurate pair-wise document similarity. We are currently working on an incremental clustering algorithm based on maintaining high cluster coherency, which looks promising and suitable for online clustering.

References

[1] K. Aas and L. Eikvil. Text categorisation: A survey. Technical Report 941, Norwegian Computing Center, June 1999.

[2] K. Cios, W. Pedrycs, and R. Swiniarski. *Data Mining Methods for Knowledge Discovery*. Kluwer Academic Publishers, Boston, 1998.

[3] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, N.J., 1992.

[4] T. Hofmann. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *Proceedings of the 16^{th} International Joint Conference on Artificial Intelligence IJCAI-99*, pages 682–687, 1999 .

[5] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. WEBSOM—self-organizing maps of document collections. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps*, pages 310–315, Espoo, Finland, June 1997.

[6] J. D. Isaacs and J. A. Aslam. Investigating measures for pairwise document similarity. Technical Report PCS-TR99-357, Dartmouth College, Computer Science, Hanover, NH, June 1999.

[7] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, N.J., 1988.

[8] M. Junker, M. Sintek, and M. Rinck. Learning for text categorization and information extraction with ILP. In J. Cussens, editor, *Proceedings of the 1^{st} Workshop on Learning Language in Logic*, pages 84–93, Bled, Slovenia, 1999.

[9] R. Kosala and H. Blockeel. Web mining research: a survey. *ACM SIGKDD Explorations Newsletter*, 2(1):1–15, 2000.

[10] S. Y. Lu and K. S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 8:381–389, 1978.

[11] U. Y. Nahm and R. J. Mooney. A mutually beneficial integration of data mining and information extraction. In *17^{th} National Conference on Artificial Intelligence (AAAI-00)*, pages 627–632, 2000.

[12] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[13] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill computer science series. McGraw-Hill, New York, 1983.

[14] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975.

[15] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

[16] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD-2000 Workshop on TextMining*, August 2000.

[17] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proceedings of the 17^{th} National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, Austin, TX, July 2000. AAAI.

[18] Y. Yang and J. P. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14^{th} International Conference on Machine Learning (ICML'97)*, pages 412–420, Nashville, TN, 1997.

[19] O. Zamir, O. Etzioni, O. Madanim, and R. M. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the 3^{rd} International Conference on Knowledge Discovery and Data Mining*, pages 287–290, Newport Beach, CA, August 1997. AAAI.

[20] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of the 21^{st} Annual International ACM SIGIR Conference*, pages 46–54, Melbourne, Australia, 1998.