

Utilizing Term Proximity Based Features to Improve Text Document Clustering

Shashank Paliwal¹, Vikram Pudi¹

¹*Center For Data Engineering, International Institute of Information Technology Hyderabad, India
paliwal@students.iiit.ac.in, vikram@iiit.ac.in*

Keywords: Text Document Clustering; Document Similarity; Term Proximity; Term Dependency; Feature Weighting.

Abstract: Measuring inter-document similarity is one of the most essential steps in text document clustering. Traditional methods rely on representing text documents using the simple Bag-of-Words (BOW) model which assumes that terms of a text document are independent of each other. Such single term analysis of the text completely ignores the underlying (semantic) structure of a document. In the literature, sufficient efforts have been made to enrich BOW representation using phrases and n-grams like bi-grams and tri-grams. These approaches take into account dependency only between adjacent terms or a continuous sequence of terms. However, while some of the dependencies exist between adjacent words, others are more distant. In this paper, we make an effort to enrich traditional document vector by adding the notion of term-pair features. A **Term-Pair** feature is a pair of two terms of the same document such that they may be adjacent to each other or distant. We investigate the process of term-pair selection and propose a methodology to select potential term-pairs from the given document. Utilizing term proximity between distant terms also allows some flexibility for two documents to be similar if they are about similar topics but with varied writing styles. Experimental results on standard web document data set show that the clustering performance is substantially improved by adding term-pair features.

1 INTRODUCTION

With a large explosion in the amount of data found on the web, it has become necessary to devise better methods to classify data. A large part of this web data (like blogs, webpages, tweets etc.) is in the form of text. Text document clustering techniques play an important role in the performance of information retrieval, search engines and text mining systems by classifying text documents. The traditional clustering techniques fail to provide satisfactory results for text documents, primarily due to the fact that text data is very high dimensional and contains a large number of unique terms in a single document.

Text documents are often represented as a vector where each term is associated with a weight. The Vector Space Model (Salton et al., 1975) is a popular method that abstracts each document as a vector with weighted terms acting as features. Most of the term extraction algorithms follow “Bag of Words”(BOW) representation to identify document terms. For the sake of simplicity the BOW model assumes that words are independent of each other but this assumption does not hold true for textual data.

Single term analysis is not sufficient to successfully capture the underlying (semantic) structure of a text document and ignores the semantic association between them. Proximity between terms is a very useful information which if utilized, helps to go beyond the Bag of Words representation. Vector based information retrieval systems are still very common and some of the most efficient in use.

Most of the work which has been done in the direction of capturing term dependencies in a document is through finding matching phrases between the documents. According to (Zamir and Etzioni, 1999), a “phrase” is an ordered sequence of one or more words. Phrases are less sensitive to noise when it comes to calculating document similarity as the probability of finding matching phrases in non related documents is low (Hammouda and Kamel, 2004). But as phrase is an ‘ordered sequence’, it is not flexible enough to take different writing styles into account. Two documents may be on same topic but due to varied writing styles there may be very few matching phrases or none in worst case scenario. In such cases, phrase based approaches might not work or at best be as good as a single term analysis algorithm.

Measuring term dependency through phrases or n-grams includes dependency only between adjacent terms. However, genuine term dependencies do not exist only between adjacent words. They may also occur between more distant words such as between “powerful” and “computers” in powerful multiprocessor computers. This work is targeted in the direction of capturing term dependencies between adjacent as well as distant terms. Proximity could be viewed as indirect measure of dependence between terms. (Beeferman et al., 1997) shows that term dependencies between terms are strongly influenced by proximity between them. The intuition is that if two words have some proximity between each other in one document and similar proximity in the other document, then a combined feature of these two words when added to the original document vector should contribute to similarity between these two documents. We also suggest a feature generation process to limit the number of pairs of words to be considered for inclusion as a feature. Cosine similarity is then used to measure similarity between the two document vectors and finally Group Hierarchical Agglomerative Clustering (GHAC) algorithm is used to cluster documents.

To the best of our knowledge, no work has been done so far to utilize term proximity between distant terms for improved clustering of text documents. The contribution of this paper is two folds. Firstly we introduce a new kind of feature called **Term-Pair** feature. A Term-Pair feature consists of a pair of terms which might be adjacent to each other as well as distant and is weighted on the basis of a term proximity measure between the two terms. With the help of different weights, we show how clustering is improved in a simple yet effective manner. Secondly, we also discuss how from the large number of such possible features, only the most important ones are selected and remaining ones are discarded.

The rest of the paper is organized as follows. Section 2 briefly describes the related work. Section 3 explains the notion of term proximity in a text document. Section 4 describes our approach to the calculation of similarity between two documents. Section 5 and 6 describe experimental results and the conclusion respectively.

2 RELATED WORK

Many Vector Space Document based clustering models make use of single term analysis only. To further improve clustering of documents and rather than treating a document as Bag Of Words, including

term dependency while calculating document similarity has gained attention. Most of the work dealing with term dependency or proximity in text document clustering techniques includes phrases (Hammouda and Kamel, 2004), (Chim and Deng, 2007), (Zamir and Etzioni, 1999) or n-gram models. (Hammouda and Kamel, 2004) does so by introducing a new document representation model called the Document Index Graph while (Chim and Deng, 2007), (Zamir and Etzioni, 1999) do so with the use of Suffix Tree. In (Bekkerman and Allan, 2003), the authors talk about the usage of bi-grams to improve text classification. (Ahlgren and Colliander, 2009), (Andrews and Fox,) analyze the existing approaches for calculating inter document similarity. In all of the above mentioned clustering techniques, semantic association between distant terms has been ignored or is limited to words which are adjacent or a sequence of adjacent words.

Most of the existing information retrieval models are primarily based on various term statistics. In traditional models - from classic probabilistic models (Croft and Harper, 1997), (Fuhr, 1992) through vector space models (Salton et al., 1975) to statistical language models (Lafferty and Zhai, 2001), (Ponte and Croft, 1998) - these term statistics have been captured directly in the ranking formula. The idea of including term dependencies between distant words (distance between term occurrences) in measurement of document relevance has been explored in some of the works by incorporating these dependency measures in various models like in vector space models (Fagan, 1987) as well as probabilistic models (Song et al., 2008). In literature efforts have been made to extend the state-of-the-art probabilistic model BM25 to include term proximity in calculation of relevance of document to a query (Song et al., 2008), (Rasolofoa and Savoy, 2003). (Hawking et al., 1996) makes use of distance based relevance formulas to improve quality of retrieval. (Zhao and Yun, 2009) proposes a new proximity based language model and studies the integration of term proximity information into the unigram language modeling. We aim to make use of term proximity between distant words, in calculation of similarity between text documents represented using vector space model.

3 BASIC IDEA

The basis of the work presented in this paper is measure of proximity among words which are common in two documents. This in turn conveys that two documents will be considered similar if they have many words in common and these words appear in

‘similar’ proximity of each other in both the documents.

3.1 Proposed Model

A **Term-Pair** feature is a feature whose weight is a measure of proximity between the pair of terms. These terms may be distant i.e. one appears after certain number of terms from other or be adjacent to each other. Since it is unclear what is the best way to measure proximity, we use three different proximity measures while using the fourth proximity measure for the purpose of normalization. All these measures are independent of other relevance factors like Term Frequency(TF) and Inverse Document Frequency(idf). We chose term as a segmentation unit i.e. we measure the distance between two term occurrences based on the number of terms between two occurrences after stop words removal. $dist(t_i, t_j)$ refers to number of terms which occur between terms t_i and t_j in a document after stop words removal.

Let D be a document set with N number of documents:

$$d_n = \{ t_1, t_2, t_3, \dots, t_m \}$$

Where d_n is the n^{th} document in corpus and t_i is i^{th} term in document d_n .

In this paper we use three kinds of proximity measures to compute term-pair weights for terms t_i and t_j :

1. $dist_{min}(t_i, t_j)$

where $dist_{min}(t_i, t_j)$ is the minimum distance expressed in terms of number of words between terms t_i and t_j in a document d. Distance of 1 corresponds to adjacent terms.

2. $dist_{avg_min}(t_i, t_j)$

where $dist_{avg_min}(t_i, t_j)$ is the average of the shortest distance between each occurrence of the least frequent term and the nearest occurrence of the other term. Suppose t_i occurs less frequently than t_j , then $d_{avg_min}(t_i, t_j)$ is the average of minimum distance between every occurrence of t_i and nearest occurrence of t_j .

3. $dist_{avg}(t_i, t_j)$

where $dist_{avg}(t_i, t_j)$ is the difference between average positions of all the occurrences of terms t_i and t_j (Cummins and O’Riordan, 2009).

Example 1: Let in a document, t_1 occurs at {2, 6, 10} positions and t_2 occurs at {3, 4} positions. Then,

- $dist_{min}(t_1, t_2) = (3-2) = 1.0$
- $dist_{avg_min}(t_1, t_2) = ((3-2) + (6-4))/2 = 1.5$
- $dist_{avg}(t_1, t_2) = ((2+6+10)/3) - ((3+4)/2) = 2.5$

4 SIMILARITY COMPUTATION BETWEEN DOCUMENTS

Computing similarity between two documents consists of three steps which are as follows :

1. For every document we form a set of highly ranked terms which are discriminative and help distinguish the concerned document from other documents.
2. We form an enriched document vector by adding term proximity based features to traditional single term document vectors.
3. We compute similarity between two documents using cosine similarity measure.

4.1 Term-Pair Feature Selection

In document retrieval framework, the most obvious choices of term pairs for measuring proximity are terms present in the query. However while calculating similarity between two documents, out of the large combinations of term pairs possible, only the most important ones should be selected. Also, not only strong dependencies are important but weak dependencies can not be neglected as they too might contribute to similarity or dissimilarity between the documents as per the case.

Due to including term dependencies between distant terms, the possible set of term pairs are ${}^m C_2$ if we assume a document d_i to have m unique words. To select only those combinations which are useful in our calculation of similarity we first sort terms of a document on the basis of tf-idf weights and then consider only highly ranked words. If this set of highly ranked words is denoted by *HRTerms* then for a pair of words to be considered for inclusion as a feature in document vector, at least one of the words must belong to this set *HRTerms*.

If the set of words which are common in both the documents is represented by *CTerms* then for a pair of terms t_i and t_j to be considered as a feature if :

1. Both t_i and t_j belong to *HRTerms*, then (t_i, t_j) is a feature (Algorithm 1).

or

1. t_i and t_j belong to *CTerms* and either t_i or t_j belongs to *HRTerms*, then (t_i, t_j) is a feature (Algorithm 2).

4.1.1 Algorithm for building First Term-Pair Feature Set FTPairs

Algorithm 1 first forms a set of highly ranked words for a document d_i consisting of all those terms

Term	Description
D	Set of Documents in a data set
d_i	i^{th} document of set D
$Dterms_i$	Set of terms belonging to d_i
$HRterms_i$	Set of Highly Ranked terms for d_i
$FTpairs_i$	Set of First Term-Pair features for d_i
$SDTpairs_{(i,j)}$	Set of Second dynamic Term-Pair features for d_i when d_j is encountered
$CTerms_{(i,j)}$	Set of terms which are common between d_i and d_j

Table 1: Notations

which have a tf-idf weight higher than a certain user-specified minimum threshold. Now, all the possible combination of pairs of terms from $HRTerms_i$ are then added to set $FTPairs$. If $HRTerms_i$ consists of k terms then $FTPairs$ will consist of ${}^k C_2$ Term-Pair Features.

Algorithm 1 : Extracting First Term-Pair Features

Input: $Dterm_i$, *Minimum Threshold*
Output: $HRTerms_i$ and $FTPairs_i$

$FTPairs_i = \{\}$
 $HRTerms_i = \{\}$

foreach term $t \in Dterm_i$
 if ($tf - idf \text{ weight}(t) \geq \text{Minimum Threshold}$)
 $HRTerms_i = HRTerms_i \cup t$
 end if
end for

$\setminus \setminus$ Let $HRTerms_i$ obtained above be $\{t_1, t_2, t_3, \dots, t_k\}$

for ($m = 1; m < k; m++$)
 for ($n = m; n \leq k; n++$)
 $FTPairs_i = FTPairs_i \cup \{<t_m, t_n>\}$
 end for
end for

4.1.2 Algorithm for building Second Dynamic Term-Pair Feature Set $SDTpairs$

Algorithm 2 extracts Term-Pair features dynamically before computing the similarity between two documents d_i and d_j . These features are primarily based

on set of common words $CTerms_{(i,j)}$ between the two documents. Although all the pair of terms from this set are candidates for Term-Pair features, only those pairs are finally added as features of whose at least one term belongs to set of highly ranked terms for respective documents. It is important to note that $SDTPairs_{(i,j)}$ is different from $SDTPairs_{(j,i)}$ since set $HRterms$ is different for both the documents.

Algorithm 2 : Extracting Second Dynamic Term-Pair Features

Input: $Dterm_i$, $Dterm_j$, $HRTerms_i$, $HRTerms_j$, $FTPairs_i$ and $FTPairs_j$
Output: $SDTPairs_{(i,j)}$ and $SDTPairs_{(j,i)}$

$SDTPairs_{(i,j)} = \{\}$
 $SDTPairs_{(j,i)} = \{\}$
 $CTerms_{(i,j)} = \{\}$
 $CTerms_{(i,j)} = Dterms_i \cap Dterms_j$
 $\setminus \setminus$ Let $CTerms_{(i,j)}$ obtained above be $\{t_1, t_2, \dots, t_l\}$

for ($m = 1; m < l; m++$)
 for ($n = m; n \leq l; n++$)
 if ($t_m \in HRTerms_i$) || ($t_n \in HRTerms_i$)
 if $<t_m, t_n> \notin FTPairs_i$
 $SDTPairs_{(i,j)} = SDTPairs_{(i,j)} \cup \{<t_m, t_n>\}$
 end if
 end if
 if ($t_m \in HRTerms_j$) || ($t_n \in HRTerms_j$)
 if $<t_m, t_n> \notin FTPairs_j$
 $SDTPairs_{(j,i)} = SDTPairs_{(j,i)} \cup \{<t_m, t_n>\}$
 end if
 end if
end for

Example 2:

Document 1 = “**Document** clustering techniques mostly rely on **single term analysis** of text.”

Document 2 = “**Traditional data mining** techniques do not work well on text document clustering.”

Considering that all of the words shown in bold belong to the set of highly ranked words for both the documents, term pairs that will be a part of document vectors as term-pair feature for Document 1 as per possible cases mentioned above are :

1. According to **Algorithm 1**: $\{T_1, T_4\}$, $\{T_1, T_5\}$, $\{T_1, T_6\}$, $\{T_4, T_5\}$, $\{T_4, T_6\}$, $\{T_5, T_6\}$.
2. According to **Algorithm 2**: $\{T_1, T_2\}$, $\{T_1, T_3\}$, $\{T_1, T_7\}$.

where T_i is the word with position i in *Document 1* (see Table 2) after stop word removal. And

Word	Positions in Document 1	Positions in Document 2
analysis	6	-
clustering	2	7
data	-	2
document	1	6
mining	-	3
single	4	-
techniques	3	4
term	5	-
text	7	5
traditional	-	1

Table 2: Words present in Document 1 and 2 and their corresponding indices in respective documents after **stop-words** removal

{**document,clustering,techniques,text**} is the set of terms which are common in both the documents.

4.2 Enriching Original Document Vector with Term-Pair Feature

So the term pairs obtained from the above step along with their respective term proximity weights (tpw) are added as features to original document vector with unigrams or single terms and their respective tf-idf weights. These weights are shown in Table 3.

Feature	Weight
Single Term	tf-idf : $\log(1 + tf_{t,d}) * \log(\frac{N}{x_t})$
Term Pair (based on Term Proximity)	$tpw_1 = \frac{dist_{min}(t_i,t_j)}{SL}$ $tpw_2 = \frac{dist_{avg_min}(t_i,t_j)}{SL}$ $tpw_3 = \frac{dist_{avg}(t_i,t_j)}{SL}$

Table 3: Feature Weighting Schema. $tf_{t,d}$ is frequency of term t in document d , N is number of documents in corpus and x_t is number of documents in which term t occurs.

We use “span length” to normalize the term-pair weights. It is the number of terms present in the document segment which covers all occurrences of common set of words between two documents. The word “span” has been used here in a similar sense as used by (Tao and Zhai, 2007), (Hawking et al., 1996). (Tao and Zhai, 2007) define two kind of proximity mechanisms namely span-based approaches and distance aggregation approaches. Span-based approaches measure proximity based on the length segment covering all the query terms and distance aggregation approaches measure proximity by aggregating pair-wise distances between query terms. Span-based approaches do not consider the internal structure of terms and calculate proximity only on the basis of text

spans. According to their definitions, $dist_{min}(t_i,t_j)$, $dist_{avg_min}(t_i,t_j)$, $dist_{avg}(t_i,t_j)$ belong to class of distance aggregation while span length is itself a proximity measure belonging to class of span-based approaches. However, we treat span length simply as a normalization factor and not a proximity measure. The reason behind this treatment of span-length as a simple normalization factor and not a proximity measure is the basic difference between query-document similarity and document-document similarity. When all the query terms appear in a small span of text, it is reasonable to assume that such a document is more relevant to query. However, query terms are generally more closely related as compared to words common between two documents. It would not be reasonable, in our opinion to assume that when set of common words between two documents appear in a small text span then this contributes to similarity between two documents. There might be very few words common between two documents and thus these words can occur in a very small text span in one or both of the documents but this does not make those two documents similar. Utilizing span-length as a normalization factor caters to above mentioned problem and helps to normalize such proximities.

For Example 2, for document 1 span-length¹ is (7-1)=6 since document is the first word and text is the last word in Document 1 which are common between the two documents. Similarly, for Document 2 span-length² is (7-4)=3.

4.3 Similarity Computation

We use cosine similarity to measure similarity between enriched document vectors. Cosine similarity between two document vectors \vec{d}_1 and \vec{d}_2 is calculated as

$$Sim(\vec{d}_1, \vec{d}_2) = \frac{\vec{d}_1 \cdot \vec{d}_2}{|\vec{d}_1| |\vec{d}_2|}$$

where (\cdot) indicates the vector dot product and $|\vec{d}|$ indicates the length of the vector \vec{d} .

¹Position of word “document” is 1 and position of “text” is 7 in Document 1.

²Position of word “techniques” is 4 and position of “clustering” is 4 in Document 2.

5 EXPERIMENTAL RESULTS AND DISCUSSION

We conducted our experiment on web dataset³ consisting of 314 web documents already classified into 10 classes.

No kind of bound has been kept on maximum number of hops between the pair of terms which combine to form a term-pair feature as the terms which are close to each other in one document but far away from each other in other documents are the ones which contribute to dissimilarity between the two documents and it is important to keep them. Experimental results also support this and unlike in relevance model, where a limit is generally kept on the distance between two terms in terms of number of words which occur between them.

To form the set *HRTerms* of highly ranked words, we sort terms on the basis of their tf-idf weights. These are the words which are discriminative and help to distinguish a document from other unrelated documents. If the average of tf-idf weight of all the words in a document is denoted by *avg*, then all the words whose tf-idf weight is greater than ($\beta \times avg$) belong to set *HRTerms*. Here β is a user-defined value such that $0 \leq \beta \leq 1$. For our experiments, we use β as **0.7**.

We use F-measure score to evaluate the quality of clustering. F-measure combines *precision* and *recall* by calculating their harmonic mean. Let there be a class *i* and cluster *j*, then *precision* and *recall* of cluster *j* with respect to class *i* are as follows:

$$Precision(i, j) = \frac{n_{ij}}{n_j} \quad Recall(i, j) = \frac{n_{ij}}{n_i}$$

where

- n_{ij} is the number of documents belonging to class *i* in cluster *j*.
- n_i is number of documents belonging to class *i*.
- n_j is the number of documents in cluster *j*.

Then F-score of class *i* is the maximum F-score it has in any of the clusters :

$$F\text{-score}(i) = \frac{2 * Precision * Recall}{Precision + Recall}$$

The overall F-score for clustering is the weighted average of F-score for each class *i* :

$$F_{overall} = \frac{\sum_i (n_i * F(i))}{\sum_i n_i}$$

where n_i is the number of documents belonging to class *i*.

³<http://pami.uwaterloo.ca/~hammouda/webdata>

For clustering we use GHAC with complete linkage with the help of a java based tool⁴. We chose traditional tf-idf weighting based single term approach as our baseline approach. We obtained F-score of **0.77** with traditional single term document vector with tf-idf weighting. We perform two different experiments using different document vectors.

5.1 Experiment 1

We add term-pair features to original document vector consisting of individual terms. It is important to note we do not apply any kind of dimensionality reduction on original document vector which consists of only single term features since our aim is to investigate whether adding term pair features to original document vectors could improve clustering or not. The F-scores using different term-pair weights are tabulated in Table 4.

Term Pair Weight	F-score	% improvement over baseline
$tpw_1(t_i, t_j) = \frac{d_{min}(t_i, t_j)}{SL}$	0.840	7.91
$tpw_2(t_i, t_j) = \frac{d_{minavg}(t_i, t_j)}{SL}$	0.81	4.08
$tpw_3(t_i, t_j) = \frac{d_{avg}(t_i, t_j)}{SL}$	0.799	2.67

Table 4: Obtained F-score and percentage improvement over baseline approach with different term pair weights.

The results of this experiment agree with experiments in relevance model where too proximity measure based on minimum pair distance generally also performs well in relevance model as reported by (Tao and Zhai, 2007) and (Cummins and O’Riordan, 2009).

5.2 Experiment 2

To further investigate the significance of Term-Pair features, we combined term-pair features vector based cosine similarity with traditional single term feature vector based cosine similarity using weighted average of two similarities. If $Sim_{tpf}(d_1, d_2)$ represents cosine similarity between document vectors consisting of only term-pair features and $Sim_{tf-idf}(d_1, d_2)$ represents cosine similarity between document consisting of only traditional single term, tf-idf weighted features, then combined similarity is given by equation (1) :

$$Sim(d_1, d_2) = \alpha * Sim_{tpf}(d_1, d_2) + (1 - \alpha) * Sim_{tf-idf}(d_1, d_2)$$

⁴Link to download tool : <http://www.cs.umb.edu/smarog/agnes/agnes.html>

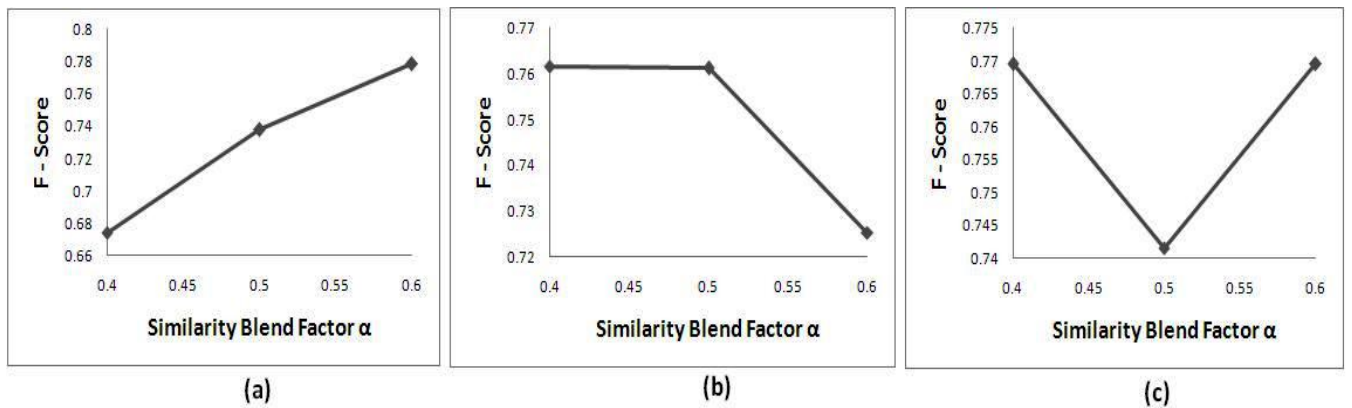


Figure 1: Variation of F-Scores for different Term-Pair weights with α . (a) For $tpw_1(t_i, t_j)$, (b) For $tpw_2(t_i, t_j)$ and (c) For $tpw_3(t_i, t_j)$

where α is the similarity bend factor and its value lies in the interval $[0,1]$. (Hammouda and Kamel, 2004)

We experimented with three values of α - 0.4, 0.5 and 0.6 for each of the three Term-Pair weights. Figure 1 shows the results with respective curves. The obtained F-Scores however, are mostly less than the baseline score. For $tpw_1(t_i, t_j)$, F-scores improve as value of α is increased from 0.4 to 0.6. Better results were obtained with Experiment 1 which suggests that Term-Pair features tend to combine well with single term features as compared to combining similarity of both the features. Thus, in other words a document is better represented by a vector consisting of both features and such a document vector helps in distinguishing one document from other documents.

We also formed a document vector which consisted only of Term-Pair features without any single term features and measured similarity between these document vectors. We can say for this experiment, α is taken as one. With $tpw_1(t_i, t_j)$ we obtained a F-Score of **0.73**. This F-score might be less than that obtained with traditional tf-idf weighted vectors, but still highlights the importance of Term-Pair features and utilizing term dependency between distant terms for clustering of documents could prove to be a new dimension for text document representation and clustering.

6 CONCLUSIONS

The presented approach might not provide best results but are definitely promising. It is to be kept in mind that the purpose of this paper is to determine whether document clustering can be improved by adding simple term proximity based Term-Pair features. There are many more possibilities such as to investigate effect models other than vector space

model, to take different similarity measure, to apply different weighting schemes for Term-Pair features. In the future, we are working on developing a model which is suitable and make full use of term proximity between distant terms. Based on the results obtained, it is our intuition that if such a simple approach can improve the clustering then a more complex and complete approach can prove to be very useful and produce much better clustering.

REFERENCES

- Ahlgren, P. and Colliander, C. (2009). Document-document similarity approaches and science mapping: Experimental comparison of five approaches. *Journal of Informetrics*, 3(1):49–63.
- Andrews, N. O. and Fox, E. A. Recent Developments in Document Clustering. Technical report, Computer Science, Virginia Tech.
- Beeferman, D., Berger, A., and Lafferty, J. (1997). A model of lexical attraction and repulsion. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics, EACL '97*, pages 373–380, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Bekkerman, R. and Allan, J. (2003). Using bigrams in text categorization.
- Chim, H. and Deng, X. (2007). A new suffix tree similarity measure for document clustering. In *Proceedings of the 16th international conference on World Wide Web, WWW '07*, pages 121–130, New York, NY, USA. ACM.
- Croft, W. B. and Harper, D. J. (1997). *Using probabilistic models of document retrieval without relevance information*, pages 339–344. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Cummins, R. and O’Riordan, C. (2009). Learning in a pairwise term-term proximity framework for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development*

- in information retrieval*, SIGIR '09, pages 251–258, New York, NY, USA. ACM.
- Fagan, J. (1987). Automatic phrase indexing for document retrieval. In *Proceedings of the 10th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '87, pages 91–101, New York, NY, USA. ACM.
- Fuhr, N. (1992). Probabilistic models in information retrieval. *The Computer Journal*, 35:243–255.
- Hammouda, K. M. and Kamel, M. S. (2004). Efficient phrase-based document indexing for web document clustering. *IEEE Trans. on Knowl. and Data Eng.*, 16:1279–1296.
- Hawking, D., Hawking, D., Thistlewaite, P., and Thistlewaite, P. (1996). Relevance weighting using distance between term occurrences. Technical report.
- Lafferty, J. and Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 111–119, New York, NY, USA. ACM.
- Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 275–281, New York, NY, USA. ACM.
- Rasolofo, Y. and Savoy, J. (2003). Term proximity scoring for keyword-based retrieval systems. In *Proceedings of the 25th European conference on IR research*, ECIR'03, pages 207–218, Berlin, Heidelberg. Springer-Verlag.
- Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18:613–620.
- Song, R., Taylor, M. J., Wen, J.-R., Hon, H.-W., and Yu, Y. (2008). Viewing term proximity from a different perspective. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval*, ECIR'08, pages 346–357, Berlin, Heidelberg. Springer-Verlag.
- Tao, T. and Zhai, C. (2007). An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 295–302, New York, NY, USA. ACM.
- Zamir, O. and Etzioni, O. (1999). Grouper: A dynamic clustering interface to web search results. pages 1361–1374.
- Zhao, J. and Yun, Y. (2009). A proximity language model for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 291–298, New York, NY, USA. ACM.