Representative Based Document Clustering

Arko Banerjee[1] and Arun K. Pujari[2]

[1] College of Engineering and Management, Kolaghat, WB, India
arko.banerjee@gmail.com
[2] University of Hyderabad, Andhra Pradesh, India
arun.k.pujari@gmail.com

Abstract. In this paper we propose a novel approach to document clustering by introducing a representative-based document similarity model that treats a document as an ordered sequence of words and partitions it into chunks for gaining valuable proximity information between words. Chunks are subsequences in a document that have low internal entropy and high boundary entropy. A chunk can be a phrase, a word or a part of word. We implement a linear time unsupervised algorithm that segments sequence of words into chunks. Chunks that occur frequently are considered as representatives of the document set. The representative based document similarity model, containing a term-document matrix with respect to the representatives, is a compact representation of the vector space model that improves quality of document clustering over traditional methods.

Keywords: document clustering, sequence segmentation, word segmentation, entropy.

1 Introduction

Document clustering is an unsupervised document organization method that put documents into different groups called clusters, where the documents in each cluster share some common properties according to a defined similarity measure. In most document clustering models similarity between documents is measured on basis of matching with single words rather than matching with phrases. The motivation of this paper is to bring the effectiveness of phrase based matching in document clustering. Work related to phrase based document clustering that has been reported in literature is limited. Zamir *et al.* [3][4] proposed an incremental linear time algorithm called Suffix Tree Clustering (STC), which creates clusters based on phrases shared between documents. They claim to achieve $nlog(n)$ performance and produce high quality clusters. Hammouda et al [5] proposed a document index model which implements a Document Index Graph that allows incremental construction of a phrase-based index of the document set and uses an incremental document clustering algorithm to cluster the document set.

In this paper we introduce a representative based document clustering model. The model involves three main phases: sequence segmentation, document representation, and clustering. Let D be a set of documents containing documents

$d_1, d_2, ..., d_r$. Sequence segmentation begins with converting each document d_i into an ordered sequence of words say s_i by removing stop-words and spaces between words. Let $S \leftarrow \bigcup_{i=1}^{r} s_i$ and $S =< e_1, e_2, .., e_N >$, where each e_i is an alphabet. Then parsing of the sequence S is performed using the entropy and frequency measures to produce a set of chunks. The first phase is an essential pre-processing method to gain valuable proximity information between words. The document representation phase begins with identifying chunks that occur frequently and are selected as representative chunks.The document set is then represented as a term-document matrix where each document is represented by a feature vector which contains metric scores such as binary score (presence or absence of a term in the document), TF (i.e., within-document term frequency) or TF.IDF with respect to the selected representatives. This phase reduces the high dimensionality of the feature space, which in turn improves the clustering efficiency and performance. In the final phase the target documents $d_1, d_2, ..., d_r$ are grouped into distinct clusters by applying clustering algorithms on the term-document matrix.To demonstrate the effectiveness of the model, we have run our method on several datasets and found promising results.

The rest of this paper is organized as follows. Section 2 describes boundary entropy and frequency as sequence segmentation measures. Section 3 explains our proposed unsupervised sequence segmentation algorithm that we implemented in the first phase of our model. Section 4 introduces the concept of representative based document clustering method with a toy example which is implemented in the second and third phases. Section 5 provides the detailed experimental evaluation of our method with other existing algorithms. Finally, some concluding remarks and directions of future research are provided in Section 6.

2 Boundary Entropy and Frequency as Segmentation Measures

A successful sequence segmentation algorithm seeks to maximize the unpredictability between subsequences or chunks. To do that, alphabet that succeeds and alphabet that precedes a chunk, should have their unpredictability maximized with respect to the chunk. The boundary entropy of a chunk is a measure that expresses the magnitude of unpredictability at its end boundary and hence we measure the boundary entropies of each chunk and its reverse chunk to predict both boundaries.

To find boundary entropy of all the chunks and reverse chunks in S of length less than equal to n, an ngram TRIE of depth $n + 1$ is generated by sliding a window along the sequence S. Let $w^1_{i,j} =< e_i, e_{i+1}, ..., e_j >$ represents a chunk of length n starting and ending at i^{th} and j^{th} position in S, respectively, where $1 \leq i < j \leq N$ and $j - i \leq n - 1$. Also let $w^2_{i,j} =< e_j, e_{j-1}, ..., e_i >$ represents the corresponding reverse chunk of the said chunk $w^1_{i,j}$. A chunk or a reverse chunk in S is represented by a node of the TRIE consisting of two frequency fields. For example, in Fig. 1 a TRIE with depth 2 is generated using the sequence $\{e\ a\ b\ c\ a\ b\ d\}$. Every chunk and the reverse chunk of length 2 or less in the sequence is represented by a node in the tree. The right and left frequency fields represent frequencies of the chunk and the reverse chunk, respectively. For example, the chunk $\{a\ b\}$ and the reverse chunk $\{b\ a\}$ occurs twice. The reverse chunk $\{a\ c\}$ occurs once but the chunk $\{a\ c\}$ does not occur. Let a node n_k represents the frequencies of a chunk and a reverse chunk in S by f_k^1 and f_k^2, respectively. If n_k has m children, then they would be denoted by $n_{k,1}, ..., n_{k,m}$ having frequencies $f_{k,1}^t, ..., f_{ki,m}^t$, respectively, where $t = 1, 2$.

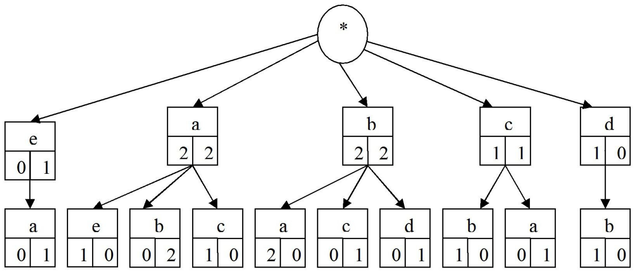

Fig. 1. Two way TRIE structure of the sequence {e a b c a b d}

The conditional probability of an alphabet succeeding/preceding a chunk is measured by the frequency of the alphabet succeeding/preceding the chunk divided by the frequency of the chunk, which we denote by $Pr(n_{kh}^t) = f_{kh}^t / f_k^t, t = 1, 2$. The boundary entropy of the chunk/reverse chunk is then the summation of entropy of the conditional probabilities of all such succeeding/preceding alphabets, which we denote by $H(n_k^t) = -\sum_{h=1}^{m} Pr(n_{kh}^t) \log Pr(n_{kh}^t)$, $t = 1, 2$. For example, the node **a** as a chunk in Fig. 1 has low entropy equal to 0 because it has only one child **b**. Hence there is a chance that $\{a\ b\}$ forms a chunk. Whereas $\{b\}$ as a chunk with high entropy equal to 1.0 has a high chance of being the end of a chunk. Therefore we expect the segmented sequence as $\{\ e\ a\ b*\ c\ a\ b*\ d\ \}$, where * denotes the end of a chunk. Again $\{b\}$ as a reverse chunk has an entropy equal to 0 hence high chance of$\{b\ a\}$ to form a reverse chunk. Whereas $\{a\}$ as a reverse chunk has an entropy equal to 1.0 hence a high chance of being the end of a reverse chunk. Therefore after combining forward and backward segmentation the expected segmented sequence should be $\{e\ \#a\ b{*}c\#a\ b{*}d\}$, where # denotes the end of a reverse segment.

By sliding a window of length n along a sequence, where n varies from 2 (chunks with one alphabet not considered) to a maximum window size, say $W(<$ depth of the TRIE), each position in the sequence receives separate $(W - 1)$ boundary entropy values for chunks and for reverse chunks. We maintain a final boundary score at each position by adding all $(W - 1)$ boundary entropy values separtely for chunks and for reverse chunks and we call them the *forward entropy score* and *backward entropy score*, respectively. Let the *forward entropy score* and *backward entropy score* are denoted by H_j^1 and H_j^2 for a position j in S, respectively. Then $H_j^t \leftarrow \sum_{1 \leq i, 0 < j-i \leq (W-1)} H(Find\_Trie\_Node(w_{i,j}^t)), t = 1, 2$

and $1 < j \leq N$. Here the $Find\_Trie\_Node$ function returns the node of the TRIE that represents $w^t_{i,j}$. We will utilize the said scores to predict word boundaries in our segmenation algorithm described in section 4. In the following we derive the frequency information of chunks in the sequence to perform segmentation.

We make an effort to determine the boundary between two consecutive chunks by comparing frequencies of the chunks with subsequences that straddle the common boundary. We explain the situation using the following example. Fig. 2 shows a sequence of ten alphabets W O R D 1 W O R D 2 that represents two consecutive words WORD1 and WORD2. The goal is to determine whether there should be a word boundary between "1" and "W". Let us assume that at an instance a window of length 10 contains the words and it considers a hypothetical boundary in the middle that is between WORD1 and WORD2. To check whether the boundary is a potential one we count the number of times the two chunks (words WORD1 and WORD2) are more frequent than subsequences inside the window of same length that straddle the boundary. Fig. 2 shows four possible straddling subsequences of length five. For example, frequency of one of the straddling subsequence 1WORD should be low compared to WORD1 or WORD2, since WORD2 has less chance of occurring just after WORD1. We maintain a score that is incremented by one only if WORD1 or WORD2 are more frequent than that of 1WORD. So, if both possible chunks are higher in frequency than a straddling subsequence, the score to the hypothetical boundary location is incremented twice. In our algorithm we consider a window of length $2n$, where n varies from 2 to the maximum length W. A window of length $2n$ has its hypothetical boundary separating alphabets at n^{th} and $(n+1)^{th}$ positions, hence number of straddling subsequences across the boundary would be $(n-1)$. Therefore, by comparing all $(n-1)$ straddling subsequences with both chunks inside the right and left windows the boundary location receives total $2(n-1)$ scores.To normalize the scores of different window length we average the scores by dividing it with $2(n-1)$. The final score at each location is then the sum of all $(n-1)$ average scores contributed by chunks inside the window having length of 2 to W. The final score at each location of the sequence, which we call the *frequency score*, measures the potentiality of the location to be a word boundary and is utilized by segmentation algorithm in section 4. Let the *frequency score* at j^{th} position in S using a window of length $2n (2 \leq n \leq W)$ is denoted by F_j. If a straddling sequence starts inside the window from the k^{th} position in S, then F_j can be written as $F_j \leftarrow \sum_{n=2}^{W}(\sum_{k=j-n+2}^{j}(\delta_{Freq(Find\_Trie\_Node(w^1_{k,k+n-1}))<Freq(Find\_Trie\_Node(w^1_{j-n+1,j}))}+$

$\delta_{Freq(Find\_Trie\_Node(w^1_{k,k+n-1}))<Freq(Find\_Trie\_Node(w^1_{j+1,j+n}))}))/2(n-1)$, where $\delta_{TRUE} = 1$ and $\delta_{FALSE} = 0$. Here $Freq$ function returns the frequency of a node in the TRIE.

In most iterations the window may partially contain two consecutive actual words of different length or it may contain one of the two actual words, where in both cases the words or part of words contained by the window contribute properly to the boundary score. For a long window the left and right windows

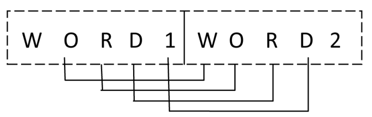

Fig. 2. Detection of word boundary by comparing frequencies of WORD1 and WORD2 with all four possible straddling subsequences of length five

would contain many consecutive actual words. In that case, like all straddling subsequences across the boundary, the consecutive words in the left and right windows would occur mostly once in the sequence. So they would contribute almost nothing to the boundary score. As window length is increased after a certain length, the contribution becomes mostly zero, which is a kind of getting rid of window-size parameter. In the following section we explain the segmentation algorithm that segments an input sequence using boundary entropy and frequency scores.

3 The Sequence Segmentation Algorithm

The segmentation method implements three segmentation scores at jth position in S namely, forward entropy score H_j^1, backward entropy score H_j^2 and frequency score F_j (derived in section 2) to perform three separate segmentations of S say, SS_{ent}^1, SS_{ent}^2 and SS_{freq}, respectively . Instead of taking threshold values from the user the method cuts the text at locations that have locally maximum segmentation scores. We implemented entropy scores and frequency scores separately to perform segmentation of sequences, and discovered that chunking is more robust if both entropy and frequency works together.Therefore the segmentation method combines all the three segmented sequences into a final consensus sequence. Cuts that are common in all three segmentations are considered to be very accurate but less in number. To get moderate number of accurate cuts the segmentation method considers cuts that are common in at least two segmentations. We will call the segmentation method the *Consensus Segmentation using Entropy and Frequency* (CSEF) algorithm. If SS is denoted as the final segmentation of the sequence S derived by CSEF, then $SS \leftarrow (SS_{ent}^1 \cap SS_{ent}^2) \cup (SS_{ent}^1 \cap SS_{freq}) \cup (SS_{ent}^2 \cap SS_{freq})$. In the following we compare CSEF with an existing sequence segmentation algorithm using a Toy example.

The CSEF algorithm is a linear time sequence segmentation algorithm that requires the maximum window length as the only input parameter. The CSEF algorithm is a robust unsupervised algorithm that works without the help of an existing lexicon and performs good even for small input document set. We compare CSEF with the Voting Experts (VE) algorithm of Cohen et al [1]. The VE algorithm is a linear time one way sequence segmentation algorithm that detects chunks by giving voting scores for each location in the sequence by maintaining a TRIE structure. In the following, we compare the performances of CSEF and VE with an example.

Input: Miller was close to the mark when he compared bits with segments. But when he compared segments with pages he was not close to the mark. his being close to the mark means that he is very close to the goal. segments may be identified by an information theoretic signatures. page may be identified by storage properties. bit may be identified by its image.
Output of VE: Mi*lle*rwas*close*tot*he*ma*rk*whe*nhe*co*mpa*re*dbi*ts* wi* thse*gmen*ts*But*whe*nhe*co*mpa*re*dse*gmen*ts*wi*thpa*ge*she*was *not*close *tot*he*ma*rkhi*sbe*ingc*lose*tot*he*ma*rkme*ans*that*he*isve* ryclose *tot*he*goa*lse*gmen*ts*ma*ybe*ide*nti*fie*dbya*ni*nfo*rmati*ont* he*ore*ti* csi *gnat*ure*spa*ge*ma*ybe*ide*nti*fie*dbys*tora*ge*pro*pe *rtie*sbi*tma*y be*ide *nti*fie*dbyi*tsi*mage
Output of CSEF: Mill*er*was*close*to*the*mark*when*he*compar*ed* bitswit h* segments*But* when*he*compa*red*segments*with*page*she*was*not*clo se* to*the*mark*his*bei*ng* close*to*the*mark*me*an*stha*the*isvery*close * to*the*goal*segments*may*beid*ent*ifi*ed by*an*inf*or*ma*tion*the*oretic *si*gn*atures*page*may*beid*ent*ifi*edby*st*or*age*prop*er* ti*es*bit*may *beid*ent*ifi*edby*itsimage

Overall the output of CSEF shows better performance than that of VE. In experimental section we show that though VE performs better when the input document is large in size, the CSEF outperforms VE over most of the benchmarks. We have also compared (but not reported) results of CSEF with another improved version of VE called Bootstrap Voting Algorithm (BVE) by Cohen et al [2]. We found that BVE does not perform better than CSEF for most of the datasets, whereas BVE suffers from higher space and time complexity due to maintaining a knowledge tree as another voting expert. In the next section we introduce the concept of *Representative based Document Clustering*, where we implement CSEF to perform document clustering and explain it with a toy example.

4 Representative Based Document Clustering

We implement CSEF algorithm to find word segmentation to perform document clustering. The frequency fields in TRIE are updated with normalized frequencies of chunks. Chunks occurring more than mean frequency are marked as representatives of the document set. Representative chunks may represent stems of important terms(words or phrases) derived unsupervisely. By sliding the representatives along each document we compute the corresponding feature vector and gradually form the term-document matrix. A clustering algorithm is implemented on the term-document matrix to group the documents $\{d_1, d_2, ..., d_r\}$ into clusters. We explain the document clustering method with a toy example. Here input documents and their outputs are separated with semicolons.

A Toy Example
Input(set of documents formed from the input of section 3): Miller was close to the mark when he; compared bits with segments. But when he; compared segments with pages he was not; close to the mark. his being close to; the mark means that he is very close to; the goal. segments may be identified by an; information theoretic signatures. page may; be identified by storage properties. Bit; may be identified by its image.;

Representatives found after chunking by CSEF:
close; to; the; mark; when; he; segments; page; may; beid;
Given K=3, Output of K-means:

Cluster1: compared bits with segments. But when he; compared segments with pages he was not; *Cluster 2*: the goal. segments may be identified by an; be identified by storage properties. bit; may be identified by its image; information theoretic signatures. page may;*Cluster 3*: Miller was close to the mark when he; close to the mark. his being close to; the mark means that he is very close to;

In the following experimental section we produce the performance of our clustering method on some large datasets.

5 Experiments

In this section, we present an empirical evaluation of our representative based document clustering method in comparison with some other existing algorithms on a number of benchmark data sets[11][10].The results of only four document datasets are recorded as for most of other datasets we got almost similar results. The first five columns of Table 1 summarize the basic properties of the data sets. To compare the performances of CSEF with VE, F_1 score[13] is used to determine quality of segmentation from both methods. The F_1 score is the harmonic mean of precision and recall and reaches its best value at 1 and worst score at 0. In the 6^{th} and 7^{th} column of Table 1 F_1 scores of VE and CSEF are recorded, respectively. The results show that CSEF achieves better performance on all the four data sets.

Table 1. Detailed description of datasets along with comparison of VE and CSEF

Data	Source	points	words	classes	F_1 score of VE	F_1 score of CSEF
Tr31	TREC	927	10128	7	0.58	0.71
Tr41	TREC	878	7454	10	0.55	0.68
Tr45	TREC	690	8261	10	0.56	0.68
re0	Reuters-21578	1504	2886	13	0.52	0.61

The datasets considered in the experiment have their class labels known to the evaluation process. To measure the accuracy of class structure recovery by a clustering algorithm we use a popular external validity measure called Normalized Mutual Information (NMI) [12]. The value of NMI equals 1 if two clusterings are identical and is close to 0 if one is random with respect to the other. Thus larger values of NMI indicate better clustering performance. We have chosen K-means and Cluto, by Ying Zhao *et al.* [9], as our document clustering algorithms. In table 2 CSEF-Cluto and CSEF-Kmeans denote our representative based clustering model with Cluto and Kmeans, respectively. In most of the cases number of representatives chosen by CSEF is half the number of the words in the document set. We compare our results with four other document clustering algorithms namely, Clustering via local Regression (CLOR)[6], Spectral clustering with normalized cut (NCUT)[7], Local learning based Clustering Algorithm (LLCA1) and its variant (LLCA2)[8]. The NMI performances of the four algorithms are taken from the paper "Clustering Via Local Regression, by Jun Sun *et al.* [6] and are verified. The output of the algorithms depends on a parameter k, which they have mentioned as neighbourhood size. By drawing NMI/k graphs they have provided the outcomes that we recorded numerically in Table 2. *max* and *avg* in Table 2 means maximum and average NMI values attended in the range of k=5 to 120, respectively. For re0 data, only the best result they have recorded which happened for k=30.

Table 2. Comparison of performances of clustering algorithms

	tr41 (max/avg)	tr45(max/avg)	tr31(max/avg)	re0 (for k=30)
LLCA1	0.63/0.62	0.61/0.55	0.53/0.5	0.3905
LLCA2	0.63/0.6	0.61/0.53	0.53/0.47	0.3847
NCUT	0.64/0.6	0.57/0.55	0.53/0.45	0.4030
CLOR	0.66/0.64	0.65/0.61	0.57/0.5	0.4302
Cluto	0.6751	0.6188	0.6410	0.3753
CSEF-Cluto	0.7390	0.7338	0.6512	0.4134
Kmeans	0.33	0.31	0.28	0.15
CSEF-Kmeans	0.38	0.38	0.28	0.22

Table 2 shows that K-means does not give satisfactory results when applied alone but with CSEF quality of results improve. Generally in practice, Cluto produces very good results compared to other algorithms. But it performs even better when associated with CSEF, which demonstrates the effectiveness of representative based similarity approach to document clustering. Here we see that CSEF-Cluto outperforms other algorithms for most of the datasets.

6 Conclusions and Future Work

In this paper, we proposed a new approach to document clustering which is based on a representative based similarity concept that uses the idea of consensus sequence segmentation. Experimental results on many data sets show that our method together with a good clustering algorithm improves the quality of the result and outperforms other well known clustering algorithms. In the future, we want to do a deeper analysis on the underlying reason for good performance of our algorithm and also to understand when it fails to give good results. We are also looking for comparing our method with other phrase-based algorithms. A more sound concept we need to develop that would resist in generating unnecessary small segments. We will also try to automatically detect optimum window size in CSEF to make it fully automatic.

References

1. Cohen, P., Adams, N., Heeringa, B.: Voting experts: An unsupervised algorithm for segmenting sequences. Journal of Intelligent Data Analysis (2006)
2. Hewlett, D., Cohen, P.: Bootstrap Voting Experts. In: IJCAI, pp. 1071–1076 (2009)
3. Zamir, O., Etzioni, O.: Web Document Clustering: A Feasibility Demonstration. In: Proc. 21st Ann. Int'l ACM SIGIR Conf., pp. 45–54 (1998)
4. Zamir, O., Etzioni, O.: Grouper: A Dynamic Clustering Interface to Web Search Results. Computer Networks 31(11-16), 1361–1374 (1999)
5. Hammouda, K., Kamel, M.: Efficient Phrase-Based Document Indexing for Web Document Clustering. IEEE Trans. Knowl. Data Eng. 16(10), 1279–1296 (2004)
6. Sun, J., Shen, Z., Li, H., Shen, Y.: Clustering Via Local Regression. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 456–471. Springer, Heidelberg (2008)
7. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
8. Wu, M., Scholkopf, B.: A local learning Approach for Clustering. In: Advances in Neural Information Processing Systems, vol. 19 (2006)
9. Zhao, Y., Karypis, G.: Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering. Machine Learning 55, 311–331 (2004)
10. Lewis, D.D.: Reuters-21578 text categorization test collection, `http://www.daviddlewis.com/resources/testcollections/reuters21578`
11. TREC: Text REtrieval Conference, `http://trec.nist.gov`
12. Strehl, A., Ghosh, J.: Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. Journal of Machine Learning Research 3, 583–617 (2002)
13. Van Rijsbergen, C.J.: Information Retrieval, 2nd edn. Dept. of Computer Science, University of Glasgow (1979)