Lexical Chains as Document Features

Dinakar Jayarajan, Dipti Deodhare
Centre for Artificial Intelligence and Robotics,
Defence R & D Organisation,
Bangalore, INDIA.
[dinakarj, dipti]@cair.drdo.in

B Ravindran
Dept. of CSE,
IIT Madras,
Chennai, INDIA.
ravi@cse.iitm.ac.in

Abstract

Document clustering and classification is usually done by representing the documents using a bag of words scheme. This scheme ignores many of the linguistic and semantic features contained in text documents. We propose here an alternative representation for documents using Lexical Chains. We compare the performance of the new representation against the old one on a clustering task. We show that Lexical Chain based features give better results than the Bag of Words based features, while achieving almost 30% reduction in the dimensionality of the feature vectors resulting in faster execution of the algorithms.

1 Introduction

Text data usually contains complex semantic information which is communicated using a combination of words. Ideally, the representation used should capture and reflect this fact in order to semantically drive the clustering algorithm and obtain better results.

The Bag of Words (BoW) (Salton et al., 1975) scheme is a very popular scheme which has been used for representing documents. But, this scheme ignores many of the linguistic and semantic features contained in text documents. This paper explores an alternative representation for documents, using *lexical chains*, which encodes some of the semantic information contained in the document. This representation results in improved performance on the clustering tasks and achieves a drastic reduction in the size of the feature space as well.

The BoW scheme was originally designed for the Information Retrieval domain (Salton, 1989) where the aim was to 'index' the document and not necessarily to model the topic distribution. This representation has since been adopted as the *defacto* document representation scheme for supervised and unsupervised learning on documents. The BoW scheme represents features as an unordered set of words contained in the document, along with their frequency count.

The BoW scheme assumes that the distribution of words in a document reflect the underlying distribution of topics and hence if the documents are grouped on the basis of the similarity of the words contained in them, it will implicitly result in a clustering based on topics. This representation, using a simple frequency count alone, does not capture all the underlying information present in the documents. Moreover, it ignores information such as position, relations and co-occurrences among the words. In addition, the feature space formed will be very huge and sparse resulting in time and space costs as well.

Lexical Chaining is a technique which seeks to identify and exploit the semantic relatedness of words in a document. It is based on the phenomenon of *lexical cohesion* (Halliday and Hasan, 1976) and works on the premise that semantically related words co-occur close together in a passage more than "just by chance". Lexical chaining is the process of identifying and grouping such words together to form chains which in turn will help in identifying and representing the topic and content of the document.

Lexical chains have been used as an intermediate representation of text for various tasks such as au-

tomatic text summarisation (Barzilay and Elhadad, 1997; Silber and McCoy, 2002), malapropism detection and correction (Hirst and St-Onge, 1997), and hypertext construction (Green, 1998). An algorithm for computing lexical chains was first given by (Morris and Hirst, 1991) using the Roget's Thesaurus (Kirkpatrick, 1998). Since an electronic version of the Roget's Thesaurus was not available then, later algorithms were based on the WordNet lexical database (Fellbaum, 1998).

We present here a two pass algorithm to compute a representation of documents using lexical chains and use these lexical chains to derive feature vectors. These lexical chain based feature vectors are used to cluster the documents using two different algorithms - *k*-Means and Co-clustering. *k*-Means is a well studied clustering algorithm widely used in the text domain. Co-clustering, also known as bi-clustering (Madeira and Oliveira, 2004), is a clustering approach which was developed in the bioinformatics domain for clustering gene expressions. Since the text domain shares a lot of characteristics (high dimensionality, sparsity, *etc.*) of gene expression data, a lot of interest has been generated recently in applying the co-clustering approaches (Dhillon et al., 2003) to the text domain with promising results. Co-clustering (Dhillon et al., 2003; Sra et al., 2004) exploits the duality between rows and columns of the document-term matrix used to represent the features, by simultaneously clustering both the rows and columns.

We compare the clustering results obtained from document features extracted using lexical chains against those obtained by using the traditional method of bag of words.

2 Lexical Chains

Lexical chains are groups of words which exhibit lexical cohesion. Cohesion as given by (Halliday and Hasan, 1976) is a way of getting text to "hang together as a whole". Lexical cohesion is exhibited through cohesive relations. They (Halliday and Hasan, 1976) have classified these relations as:

1. Reiteration with identity of reference
2. Reiteration without identity of reference
3. Reiteration by means of super ordinate
4. Systematic semantic relation
5. Non systematic semantic relation

The first three relations involve reiteration which includes repetition of the same word in the same sense (*e.g.*, car and car), the use of a synonym for a word (*e.g.*, car and automobile) and the use of hypernyms (or hyponyms) for a word (*e.g.*, car and vehicle) respectively. The last two relations involve collocations *i.e*, semantic relationships between words that often co-occur (*e.g.*, football and foul). Lexical chains in a text are identified by the presence of strong semantic relations between the words in the text.

Algorithms for building lexical chains work by considering candidate words for inclusion in the chains constructed so far. Usually these candidate words are nouns and compound nouns. Lexical Chains can be computed at various granularities - across sentences, paragraphs or documents. In general, to compute lexical chains, each candidate word in the sentence/paragraph/document is compared, with each lexical chain identified so far. If a candidate word has a 'cohesive relation' with the words in the chain it is added to the chain. On the other hand, if a candidate word is not related to any of the chains, a new chain is created for the candidate word. Thus a lexical chain is made up of a set of semantically related words. The lexical chains obtained are then evaluated based on a suitable criteria and the better chains are selected and used to further processing. Naturally, the computation of lexical chains is predicated on the availability of a suitable database which maps relations between words.

Several algorithms have been proposed for computing lexical chains. Prominent among them are those by (Hirst and St-Onge, 1997; Barzilay and Elhadad, 1997; Silber and McCoy, 2002; Jarmasz and Szpakowicz, 2003). Except for the one by Jarmasz and Szpakowicz, all others use WordNet (Fellbaum, 1998) to identify relations among words. A brief overview of these algorithms is given in (Jayarajan et al., 2007).

WordNet is a lexical database which organises words into synonym sets or *synsets*. Each synset contains one or more words that have the same meaning. A word may appear in many synsets, depending on the number of senses that it has. The

synsets are connected by links that indicate different semantic relations such as generalisation (hypernyms), specialisation (hyponyms), part relations (holonyms[1] and meronyms[2]), *etc*.

Our approach to computing lexical chains differs from those listed above and is described in the next section.

3 Lexical Chains based Feature Vectors

All the algorithms mentioned in the previous section, try to disambiguate the sense of the word as part of the chaining process. Both Word Sense Disambiguation (WSD) and lexical chaining are very profound processes. The aim of computing the lexical chains here is to try and identify the topics in a document. If WSD has be performed as implicit step in the lexical chain computing algorithm, it tends to deteriorate the outcome of both. We feel that the words should be disambiguated by looking at their context in a sentence/paragraph as a whole. As such, we propose to perform WSD as a preprocessing step, before the word is considered for lexical chaining. We use an algorithm by (Patwardhan et al., 2003) to disambiguate the senses of the words in reference to Wordnet. We then filter out all non-noun words identified in the WSD stage. This is based on the assumption that nouns are better at reflecting the topics contained in a document than the other parts of speech. The result is a set of nouns which appear in the text along with its sense. We refer to these as 'candidate words'.

Our algorithm is based on the WordNet Lexical Database. WordNet is used to identify the relations among the words. We use only the identity and synonymy relations to compute the chains. A word has a identity or synonymy relation with another word, only if both the words occur in the same synset in Wordnet. Empirically, we found that usage of only these two relations, resulted in chains representing crisp topics.

A lexical chain contains a list of words which are related to each other and is identified using a unique numeric identifier. Each word in turn is represented as a 4-tuple <term, pos, sense, rel>, where 'pos' is the part-of-speech of the term, 'sense' is the Wordnet sense number and 'rel' is the relation of this word to the chain. In this case, we treat the two relations - identity and synonymy, as a single relation and hence this is uniformly 'IS' for all the words.

Definition 1 *Length of a lexical chain L is defined as the number of words in the chain.*

$$length(L) = \text{Number of Words in Chain L} \quad (1)$$

The length of a lexical chain is an indicator of the strength of the chain in representing a topic. Dominant topics/information will have long chains, while stray information will form extremely short chains. Each occurrence of a word in a document, will increase the length of the chain by one. Thus, the length of a chain gives a composite measure of the number of documents in which the chain occurs and the number of occurences of words in the chain in these documents.

3.1 Feature Vector Computation

We use a two pass algorithm to generate feature vectors based on lexical chains. Our algorithm works by maintaining a global set of lexical chains, each of which represents a topic. Initially, the global list is empty. In the first pass we identify all possible lexical chains for that document. This is achieved by comparing the candidate words of each document with the global list to identify those chains with which it has a identity or synonymy relation. If no chains are identified, a new chain is created and put in the global list. The candidate word is then added to the chain. At the end of this pass, we obtain a global set which lists all the chains contained in all the documents. The algorithm is presented in Algorithm 1.

In the second pass we select a subset of chains from the global set, which can be used to represent the document. We define and use a measure to evaluate and select the chains as follows:

Definition 2 *The significance of a lexical chain L in a Global set G is defined as*

[1] part of, member of, substance of relations, *e.g.*, 'wheel' is part of a 'vehicle'

[2] has part, has member, has substance relations, *e.g.*, 'wheel' has part 'rim'

Algorithm 1 Identify Chains

```
1: Maintain a global set of lexical chains, initialised to a Null set
2: for each document do
3:    for each candidate word in document do
4:       Identify lexical chains in global set with which the word has a identity/synonym relation
5:       if No chain is identified then
6:          Create a new chain for this word and insert in global set
7:       end if
8:       Add word to the identified/created chains in Global Set
9:    end for
10: end for
```

Algorithm 2 Select Chains and Generate FV

```
   for each document do
2:    Initialise feature vector to zero
      for each candidate word in document do
4:       Identify lexical chains in global set with which the word has a identity/synonym relation
      end for
6:    Compute threshold for document
      for each identified chain in global set do
8:       if utility of chain greater than threshold then
            Set component corresponding to chain in feature vector to 1
10:      end if
      end for
12: end for
```

$$sig(L) = -\frac{length(L)}{\sum\limits_{l\ \epsilon\ G} length(l)} . \log_2 \frac{length(L)}{\sum\limits_{l\ \epsilon\ G} length(l)} \tag{2}$$

The significance of a chain L measures how randomly the chain appears in the global set G. This measure helps in identifying good chains from weak, random ones in the global set.

Definition 3 *A candidate word W is related to a lexical chain L if W has an identity or synonym relation with L.*

$$\begin{aligned} related(W, L) &= 1,\ W\ and\ L\ are\ related \\ &= 0,\ otherwise \end{aligned} \tag{3}$$

Definition 4 *The utility of a lexical chain L to a document D is defined as*

$$util(L, D) = sig(L). \sum_{all\ w\ \epsilon\ D} related(w, L) \tag{4}$$

The utility of a chain L is a measure of how good L will be in representing the document. This is based on the observation that long chains are better than short ones. This measure will prefer 'good' chains from the global set, which are related to a large number of candidate words in the document.

We select and assign to the document all those chains which cross a threshold on the utility of the chain. Empirically, we found that using a threshold of 'half the average' utility for a document gave good results. For a document D, let the set of all lexical chains assignable to D be $G' \subset \text{GlobalSet G}$. The threshold for D is computed as

$$threshold(D) = \frac{\sum\limits_{l\ \epsilon\ G'} util(l, D)}{2\ .\ |G'|} \tag{5}$$

The lexical chains in the global list form the components of the feature vectors. We use a binary valued scheme, where in we put a 1 corresponding to a chain if the chain is assigned to the document and 0 otherwise. Essentially, what we obtain here is a feature vector of size equal to the number of lexical chains in the global list. The second pass of the algorithm is listed in Algorithm 2.

Cluster	Document Ids
college atheists	53675, 53357, 53540
amusing atheists and anarchists	53402, 53351
islam & dress code for women	51212, 51216, 51318

Table 2: Example of the classes obtained from grouping the documents using the subject line

4 Experiments

We use the 20 Newsgroups (Rennie, 1995) dataset to evaluate the utility of representing documents using the lexical chains (*lexchains*) scheme. The 20 Newsgroups (20NG) corpus is a collection of usenet messages spread across 20 Usenet groups. These messages are written by a wide population of net users and represent a good variation in writing styles, choice of words and grammar. Thus, we feel that the 20NG is a representative corpus for the purpose. We derive three datasets from three distinct groups of the 20NG corpus - *comp.windows.x* (`cwx`), *rec.motorcycles* (`rm`) and *alt.atheism* (`aa`). The statistics of the datasets is given in Table 1.

The documents in each dataset are further grouped on the basis of their subject lines. This grouping into classes is used as the answer key for evaluating the clustering algorithms. An example of the groups formed for the `aa` dataset is shown in Table 2.

We prepared the dataset for feature extraction by removing the complete header including the subject line and used only the body portion of the messages to compute the features. We extracted features on this cleaned data using both the BoW and *lexchains* scheme. For the BoW scheme, we first tokenised the document, filtered out the stopwords using the list obtained from (Fox, 1992) and further stemmed them using a Porter Stemmer (Porter, 1980). The feature vectors were then computed using the *tf.idf* scheme. We refer to the feature vectors thus obtained as `cwx-BoW`, `rm-BoW` and `aa-BoW` for the `cwx`, `rm` and `aa` datasets respectively. *Lexchains* based features were derived as described in Section 3.1 and are analogously referred to here as `cwx-lc`, `rm-lc` and `aa-lc`. This results in a total of six datasets. The dimensions of the feature vectors obtained are summarised in Table 3. It can be noted that the size of the feature vectors are reduced by more than 30% with the *lexchains* based features.

These six datasets were clustered using the *k*-Means and Co-clustering algorithms. The *k*-Means implementation in Matlab was used and *k* was set to 649, 340 and 196 for `cwx`, `rm` and `aa` respectively and reflects the number of classes identified in the answer key (*ref.* Table. 1). The co-clustering experiments were done using the Minimum Sum-Squared Residue Co-clustering algorithm (Sra et al., 2004) with the number of row clusters set to the same values as given to the *k*-Means algorithm.

We use a normalised edit distance based measure to evaluate the goodness of the clusters. This measure is a variant of the one used by (Pantel and Lin, 2002), which defines an edit distance as the number of merge, move and copy operations required to transform the resulting clusters to the answer key. Initially, if there are c classes in the answer key, we create c empty clusters. The measure then merges each resulting cluster to the cluster in the answer key with which it has maximum overlap, breaking ties randomly. Thus, the merge operation attempts to bring the obtained clusters as close as possible to the answer key as a whole. Subsequently, the move and copy operations are used to move (copy) the docu-

Collection	# Classes	# Documents
`comp.windows.x`	649	980
`alt.atheism`	196	799
`rec.motorcycles`	340	994

Table 1: Dataset Statistics

	BoW	*lexchain*	Reduction
`cwx`	12767	4569	64%
`aa`	8881	5980	32%
`rm`	8675	5288	39%

Table 3: Dimensionality of the Feature Vectors

	k-Means	Co-cluster	Time (secs)
cwx-BoW	203 (0.21)	**140 (0.14)**	1529
cwx-lc	**179 (0.18)**	158 (0.16)	**201**
aa-BoW	85 (0.11)	110 (0.13)	869
aa-lc	**60 (0.08)**	**82 (0.10)**	**221**
rm-BoW	**113 (0.11)**	208 (0.21)	1177
rm-lc	127 (0.12)	**144 (0.14)**	**229**

Table 4: Edit distance between obtained clusters and answer key. Normalised edit distances are given in parenthesis. The fourth column gives runtime for the co-clustering algorithm, averaged over four runs. (*For all cases, lower is better.*)

ments around so that they finally match the answer key.

We observed that the merge operation would inevitably add as many clusters as there are in the answer key to the final count, skewing the results. Hence, we define the edit distance as only the number of move and copy[3] operations required to convert the obtained clusters to that of the answer key. In effect, it measures the number of documents which are misplaced with respect to the answer key. The obtained edit distance is normalised by dividing it with the number of documents in the dataset. This will normalise the value of the measure to range between 0 and 1. The lower the value of this measure, the closer the obtained clustering is to the answer key.

The results are enumerated in Table 4. The *lexchains* based document feature gives an improvement of upto 33% over the BoW representation while achieving a reduction in dimensions of the feature vectors by more than 30% (*ref.* Table 3). We performed run time studies on the dataset using the co-clustering algorithm. The runtimes are averaged over four runs. It can be seen that a speedup of more than 74% is achieved with the *lexchain* based features[4].

Thus, the results show that the running time of the clustering algorithms is drastically reduced while maintaining or improving the clustering performance through the use of *lexchain* based features.

4.1 Discussion

A document is not just a bunch of loose words. Each word in a document contributes to some aspect of the overall semantics of the document. Classification and clustering algorithms seek to group the documents based on its semantics. The BoW scheme inherently throws away a lot of information, which would have otherwise been useful in discerning the semantics of the document. The BoW representation fails to capture and represent these semantics resulting in a less accurate representation for the documents. This fact is reflected by higher edit distance in the case of BoW based clustering in Table 4.

Earlier, Hatzivassiloglou, *et. al.* (Hatzivassiloglou et al., 2000) had studied the effects of linguistically motivated features on clustering algorithms. They had explored two linguistically motivated features - noun phrase heads and proper names and compared these against the bag of words representation. They had reported that the BoW representation was better than linguistically motivated features. We believe that noun phrase heads and proper names are inadequate representations of the semantics of a document and a more composite representation is required to obtain better results on semantically oriented tasks.

Lexical chains appear to be capable of doing this to a certain extent. During the process of computing and selecting the lexical chains, we are implicitly trying to decode the semantics of the documents. Lexical chains work on the basic premise that a document describes topics through a combination of words and these words will exhibit a cohesion among them. This cohesion can be identified using a resource such as WordNet. In the process, lexical chains capture some amount of the semantics contained in the documents, resulting in a better performance in subsequent processing of the documents.

5 Conclusion

We have shown that semantically motived features, such as lexical chains, provide a better representation for the documents, resulting in comparable or

[3]The copy count will be included only in the case of overlapping clusters, which happens if a document is in more than one cluster.

[4]It was observed empirically that the time required to compute both the BoW and *lexchain* features are nearly the same and hence can be ignored.

better performance on clustering tasks while effecting a drastic reduction in time and space complexity.

Even though the lexical chains manage to represent the semantics to a certain extent, we feel it can be further enhanced by more involved processing. A comparision of lexical chains based representation with other document representation schemes such as LSA also warrants investigation.

References

Regina Barzilay and M. Elhadad. 1997. Using lexical chains for text summarization. In *In Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL, Madrid, Spain.*

Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. 2003. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, pages 89–98.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Christopher Fox, 1992. *Information retrieval: data structures and algorithms*, chapter Lexical Analysis and Stoplists. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Stephen J Green. 1998. Automatically generating hypertext in newspaper articles by computing semantic relatedness. In D.M.W. Powers, editor, *NeMLaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language*.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.

Vasileios Hatzivassiloglou, Luis Gravano, and Ankineedu Maganti. 2000. An investigation of linguistic features and clustering algorithms for topical document clustering. In *SIGIR 2000*, pages 224–231.

Graeme Hirst and David St-Onge. 1997. Lexical chains as representation of context for the detection and correction of malapropisms. In C. Fellbaum, editor, *WordNet:An electronic lexical database and some of its applications.* The MIT Press, Cambrige, MA.

Mario Jarmasz and Stan Szpakowicz. 2003. Not as easy as it seems: Automating the construction of lexical chains using roget's thesaurus. In *Proceedings of the 16th Canadian Conference on Artificial Intelligence*.

Dinakar Jayarajan, Dipti Deodhare, B Ravindran, and Sandipan Sarkar. 2007. Document clustering using lexical chains. In *Proceedings of the Workshop on Text Mining & Link Analysis (TextLink 2007), Hyderbad, INDIA*, January.

B Kirkpatrick. 1998. *Roget's Thesaurus of English Words and Phrases*. Penguin.

Sara C. Madeira and Arlindo L. Oliveira. 2004. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

Patrick Pantel and Dekang Lin. 2002. Document clustering with committees. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 199–206, New York, NY, USA. ACM Press.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2003. Using semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics (CiCLING-03), Mexico City, Mexico.*

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*.

Jason Rennie. 1995. 20 Newsgroups dataset. Online: http://people.csail.mit.edu/jrennie/20Newsgroups/.

Gerard Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Gerard Salton. 1989. *Automatic Text Processing – The Transformation, Analysis, and Retrieval of Information by Computer*. Addison–Wesley.

H. Gregory Silber and Kathleen F. McCoy. 2002. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 28(4):487–496.

Suvrit Sra, Hyuk Cho, Inderjit S. Dhillon, and Yuqiang Guan. 2004. Minimum sum-squared residue coclustering of gene expression data. In *Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004*.