Document Clustering using Lexical Chains

Dinakar Jayarajan[1*], Dipti Deodhare[1**], B. Ravindran[2***], and Sandipan Sarkar[1†]

[1] Centre for Artificial Intelligence & Robotics, DRDO, Bangalore.
[2] Dept. of Computer Science & Engg., IIT Madras, Chennai.

Abstract. Document Clustering is an unsupervised categorisation of documents based on the contents of the documents. Numerous algorithms have been proposed for this and invariably, all of these algorithms use some variation of the vector space model to represent the documents on which the clustering algorithm is run. Lexical chains are groups of words which exhibit a cohesion among them. It is based on the idea that semantically related words co-occur more than "just by chance". By identifying and using lexical chains, it would be possible to represent the document in terms of its concepts. We propose here the use of lexical chains as an alternative representation for the documents and further leverage them to cluster the documents.

1 Introduction

Document clustering is an unsupervised categorisation of documents based on its contents. Document clusters are useful for various tasks such as text mining, topic detection and tracking, organising web search results, *etc.* Various algorithms and approaches [1] have been proposed for clustering in general, most of which can be applied to documents as well. Almost all these algorithms use the Vector Space Model [2] or a variation of it to represent these documents as data points in a very high dimensional space.

The algorithms usually work by using a similarity (or a dissimilarity) measure to compute the similarity (or dissimilarity) between the data points. Data points are then grouped such that the similarity between the points in the same cluster is very high, while it is low between points in different clusters. Typically, in document clustering, the feature vector used is the set of terms occurring in that document along with its frequency, suitably normalised.

One of the key problems in document clustering techniques is that most clustering algorithms typically require the number of clusters to be specified as an input parameter to the algorithm. It is difficult to predict the actual number of clusters among the documents and is usually just a guess. Moreover most algorithms are capable of placing the document in only one cluster. Documents

* dinakar.jayarajan@gmail.com
** dipti.deodhare@gmail.com
*** ravi@cs.iitm.ernet.in
† sarkarsandipan11@gmail.com

usually describe one or more allied topics and putting them into crisp clusters serves only to partition the dataset in groups but may not have any bearing on the underlying topics of the document. Ideally, there should be as many clusters as there are topics in the document collection, and it should be possible to place a document in as many clusters as it belongs to.

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* [3] 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 automatic text summarisation [4, 5], malapropism detection and correction [6], and hypertext construction [7]. An algorithm for computing lexical chains was first given by Morris and Hirst [8] using the Roget's Thesaurus. Since an electronic version of the Roget's Thesaurus was not available, later algorithms were based on the WordNet lexical database [9].

Morris and Hirst [8] showed that the lexical chains obtained from a document will reflect the discourse structure of the document *i.e.*, they will reflect the pattern of topics and subtopics in a document. It is this property of lexical chains that we exploit to cluster these documents. In the course of this paper, we discuss an approach to topic driven document clustering using lexical chains.

2 Lexical Chains - An Overview

Lexical chains are groups of words which exhibit lexical cohesion. Cohesion as given by Halliday and Hasan [3] is a way of getting text to "hang together as a whole". Lexical cohesion is exhibited through cohesive relations. They [3] 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 include 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 that it is made up of.

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. Several algorithms have been proposed for computing lexical chains. Prominent among them are those by [6, 4, 5, 10]. Except for the one by Jarmasz and Szpakowicz, all others use WordNet [9] to identify relations among words.

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-of (holonyms and meronyms), *etc.*

Hirst and St-Onge [6] qualify relations into extra-strong (identity and synonymy), strong (hypernymy and hyponymy) and medium strong (hypernymy, hyponymy path). Given a word, they try to identify a relation with a chain in the following order of importance - *extra strong*, *strong* and then *medium strong*. They employ a greedy strategy to disambiguate between the chains *i.e.*, words are added to the chain with which they have the strongest relation.

Barzilay and Elhadad [4] showed that chain selection cannot be done through a greedy approach. They proposed using the "whole picture" of the chain distribution in order to select the right chain. In their approach they create a component which is a list of interpretations which are exclusive of each other. As each new word is encountered, it is assessed to determine if there exists a relation with any of the existing components, failing which a new component is created. If a relation exists, all possible interpretations in the matching component are created. They then compute the score of an interpretation using an empirically derived scoring scheme based on the number and weight of the relations between the chain members. They define the best interpretation as that interpretation with the most connections. Since in this approach the number of interpretations can grow rapidly, they maintain a threshold beyond which they prune out the weak interpretations.

Silber and McCoy [5] propose a two pass algorithm for computing lexical chains. They propose four relations - identity, synonymy, hypernym/hyponym relation and sibling hypernym/hyponym tree relation. They employ a part-of-speech tagger to identify the noun instances within the document. In the first step, for each noun instance encountered, each sense of the noun instance is placed into every "meta-chain" for which it has an identity, synonym or hypernym relation with that sense. These meta-chains represent every possible interpretation of the text. In the second step, they find the best interpretation by taking each noun and looking at all its meta-chains. A score is computed based on the type of relation and distance factors to identity which meta-chain the nouns contribute the most to and then delete the noun from the other meta-chains.

Jarmasz and Szpakowicz [10] use a computerised version of the 1987 edition of Penguin's Roget's Thesaurus of English Words and Phrases [11] called the Electronic Lexical Knowledge Base (ELKB) [12] to build lexical chains. Their algorithm builds proto-chains, a set of words linked via thesaural relations. They

then score the proto-chains using a scheme similar to that of Silber and McCoy [5] to obtain the final lexical chains.

3 Clustering using Lexical Chains

We extend the notion of computing lexical chains to beyond that of just a document. We work on the premise that, instead of just computing the lexical chains with respect to a single document, if we compute the chains across many documents, we would be able to obtain a grouping of the documents based on their underlying topics and subtopics.

All algorithms described in the previous section, assume to disambiguate the sense of the word as part of the chaining process. Our experiments have revealed that though this is true, it is not good enough to be used in practise. We feel that the words should be disambiguated by looking at its context in a sentence/paragraph as a whole. Both Word Sense Disambiguation(WSD) and lexical chaining are very profound processes and tying them up tends to deteriorate the outcome of both. 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.*, [13] to disambiguate the senses before using our algorithm.

We preprocess the documents by running it through a tokeniser, a stop word removal tool, and a WSD utility [14]. We then filter out all non-noun words identified in the WSD stage. The result is a sequence of nouns which appear in the text along with its sense. We refer to these as 'candidate words'. The complete steps of the preprocessing stage is shown in Algorithm 1.

Algorithm 1 Preprocessing

```
1: for each document do
2:     Tokenise
3:     Remove stopwords
4:     Perform WSD
5:     Filter all non-nouns
6: end for
```

We base our algorithm 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. Empirically, we found that using only these two relations, resulted in chains representing crisp topics.

Our algorithm works by maintaining a global set of lexical chains, each of which represents a topic. When a new document is encountered, we preprocess it using Algorithm 1 to obtain the set of candidate words.

We now compute the lexical chains corresponding to each of the candidate words by looking up the synset for the word from WordNet. We then traverse the global list of lexical chains to identify those chains with which it has a identity or

Algorithm 2 Clustering Using Lexical Chains

```
1: Maintain a global set of lexical chains
2: for each document do
3:    for each candidate word in document do
4:       Obtain WordNet Synset for word and sense
5:       Identify lexical chains with which the word has a identity/synonym relation
6:       if No chain are identified then
7:          Create a new potential chain for this word
8:       end if
9:       Add identified/created chains to Potential Chain Set
10:   end for
11:   for each Potential Chain in Potential Chain Set do
12:      Apply chain selection heuristics
13:      if chain is retained then
14:         Add document to this chain
15:      end if
16:   end for
17: end for
18: for each document d do
19:   Initialise a new cluster
20:   for all documents where document ≠ d do
21:      Compute number of common lexical chains
22:      if number of common lexical chains ≥ threshold then
23:         Add this document to the cluster
24:      end if
25:   end for
26:   if cluster is empty then
27:      delete cluster
28:   end if
29: end for
```

synonymy relation. We refer to these identified lexical chains as potential chains. If the candidate word has no identity/synonymy relation with any of the chains in the global list, a new potential chain is created.

Once the set of all potential chains are identified, we then evaluate the potential chains, in order to select a subset of chains to which the document is added. We use a simple chain selection heuristic as follows:

$$h = \frac{\sum \text{length of all potential chains}}{\text{total no of potential chains}}$$

We select all those potential chains whose length exceeds h.

Each document is now represented by a subset of lexical chains from the global set. We now group together the documents based on the number of common lexical chains among them, to obtain a grouping which will best reflect the distribution of topics contained in the documents. An intersection of the set of lexical chains, *i.e.*, the number of common lexical chains, representing two documents will reveal how similar they are. We threshold the cardinal of the

intersection to decide if the two documents are similar. We set the threshold such that a document gets selected if it has atleast 50% common lexical chains more than number of chains assigned to the original document. The complete algorithm is listed in Algorithm 2.

Our algorithm works on the assumption that lexical chains represent the topic of the document, and grouping documents based on these lexical chains results in the documents being clustered based on its topic. *It is straightforward to see that our algorithm is capable of placing each document in multiple clusters and also automatically discovering the actual number of clusters in a collection of documents.*

4 Evaluation

We use a small dataset consisting of 31 documents derived from the 20 Newsgroups (20NG) dataset [15]. We keep the number of documents small in order to be able to do a qualitative analysis of the clusters formed as opposed to a quantitative one. To the best of our knowledge, no such qualitatively analysed benchmarks are available wherein the clustering obtained is assessed semantically with reference to topics discussed.

We selected 4, 9, 9, 5, 4 documents from *comp.graphics*, *talk.politics.guns*, *talk.mideast*, *talk.religion.misc* and *rec.auto* groups respectively of the 20NG dataset. From each group, we first picked documents at random, and in the case of some of those documents we picked up a few more documents which are related directly and indirectly to it. This was done to obtain a controlled set of clustered documents. We had also introduced a few documents to act as outliers.

We ran our algorithm on this dataset. We used an in-house implementation of a tokeniser and stop word removal tool. For the stop words, we used the list given in [16]. The WSD was done using the WordNet SenseRelate tool [14] using default parameters. WordNet v2.0 was used as the lexical database and the WordNet C API for interfacing with WordNet. The various clusters obtained along with the filenames are given in Table 1.

A recent study [17] of various document clustering algorithms revealed that the bisecting k-Means algorithm gives best result. We compare our results with that obtained by clustering the same dataset using the bisecting k-Means algorithm. We used the Cluto software [18] for the purpose and ran the algorithm with k set to 10, 16 and 26. The results are shown in Table 2.

As pointed out in the introduction, conventional clustering algorithms result in hard clusters, *i.e*, each data point can be placed in only a single cluster. Also, identifying the right number of clusters can be a problem. As already mentioned, our algorithm is capable of detecting the actual number of clusters automatically, based on the topics contained in the documents. Our algorithm discovered 26 clusters in this dataset. As can be seen from Table 2, at $k = 26$, k-Means hardly identified any clusters. At $k = 16$, which is the number of clusters identified manually, the clusters formed are reasonable.

Cluster	Document Id
1	101557
2	101574, 101597
3	101677
4	37261
5	38400
6	38406
7	38460, 38400
8	53294
9	53294, 53354
10	53294, 53354, 54152, 54455
11	54206, 53294
12	54253, 54269, 54358, 54819, 75414
13	54269, 54253
14	54819, 53294, 54253, 54269, 54358, 75414
15	75414
16	75414, 75933
17	76099, 76486
18	76184
19	76227
20	76289
21	76306
22	76506, 75933, 76184
23	82781
24	82782, 82783, 82784, 82785
25	82783
26	82784, 82785, 82783, 82782, 82781, 75414

Table 1. Clusters obtained through our algorithm. The numbers in the second column refer to file names in the 20 Newsgroups dataset.

4.1 Discussion

Most of the popular clustering algorithms such as k-Means and HACs work on the premise that the topographical distance between documents indicates similarity. Meaning, if the documents are close together in the topographical space, it implies that the documents have similar topics. But distance between points need not necessarily indicate semantic relatedness.

Another issue is with the minimum support required for these algorithms to function properly. Two words are semantically related is inferred indirectly from the fact that the words co-occur in many documents. Most of these algorithms need a significant number of input samples to identify the clusters. Our algorithm is different in that, it is independent of the number of input samples required. Even with very few samples, we can effectively identify the clusters. Moreover, semantic relatedness is inferred through word meanings and the domain. Therefore even if semantically related words co-occur only in one or very

Document Id	$k = 10$	$k = 16$	$k = 26$
101557	5	4	5
101574	8	12	1
101597	8	12	20
101677	8	7	11
37261	7	1	2
38400	7	3	4
38406	0	0	0
38460	7	2	3
53294	4	6	9
53354	5	5	8
54152	9	9	23
54206	9	15	17
54253	4	10	24
54269	4	10	24
54358	9	15	21
54455	9	9	23
54819	4	10	24
75414	9	15	16
75933	6	14	25
76099	2	11	6
76184	6	14	15
76227	6	14	7
76289	1	8	22
76306	1	8	22
76486	2	11	14
76506	6	14	25
82781	3	13	19
82782	3	13	13
82783	3	13	10
82784	3	13	12
82785	3	13	18

Table 2. Result obtained from k-Means for various values of k . The numbers in the first column refer to file names in the 20 Newsgroups dataset, and the subsequent columns indicate the cluster ids assigned to the document.

few documents, the topical overlap in the documents will be captured by the lexical chains.

Let us now examine the performance of our algorithm in detail. Consider clusters 4, 5, 6, and 7 in Table 1. Documents 37261, 38400 and 38460 are all 'Call for Presentations/Papers' type documents. Document 37261 is on 'Navy Scientific Visualization and Virtual Reality', while documents 38400 and 38460 are in the field of 'Digital Imaging' and 'Imaging' respectively. Document 38406 contains a request for a graphics package to generate graphs. Our algorithm has correctly placed documents 37261, 38400 and 38406 in distinct clusters, and

document 38460 is grouped with 38400 as the two speak about very closely related topics. Since the algorithm has seen only a small number of documents, we feel that this grouping is quite good. Ideally, there should have been another cluster which grouped all the 'Call for Papers' type documents.

A look at the results provided by bisecting k-Means in Table 2 reveals at k = 16 and 26 levels it fails to identify any grouping for these documents. At k = 10 level some grouping is shown but it is just a monolithic group of all the 'Call for Papers' type documents, which is not good enough.

Another interesting property can be observed in cluster 26. All the five documents (82781-82785) relate to 'Israel', 'Jews', and other related topics. Document 75414 also speaks about the same concepts but in a different context. Our algorithm is able to identify and group such relations, something the bisecting k-Means fails to detect altogether.

An alternative to the k-Means algorithm would have been a hierarchical agglomerative clustering(HAC) algorithm. One could argue that HAC algorithms also do not need the number of clusters to be specified in advance. But in some sense this is not true! Even HACs require the user to decide the level at which the dendrogram should be cut to obtain the clusters. Also, document feature vectors are of extremely high dimensions and HAC algorithms have quadratic time complexity. As a result, HAC algorithms will be too slow [19] for document clustering. Another drawback of the HAC algorithms is that a misjudgement in an early iteration of the algorithm will seriously affect all the further iterations. We believe our algorithm is immune to such problems.

We traced our algorithm to evaluate the performance of each stage and in the process identified certain problems. Proper nouns were getting wrongly looked up in WordNet. For example, in document 101574 the chains formed are:

1. bent, hang, knack
2. design, designing
3. station
4. waggon, wagon
5. tercel, tercelet, tiercel
6. corolla
7. architect, designer
8. sense, signified

Of the above, chains 5 and 6 are in reference to 'a male hawk' and 'a corolla of a flower' while in the document it refers to a 'Toyota Tercel Car' and 'Toyota Corolla Car' respectively. Obviously, there seems to be a problem with trying to handle proper nouns through WordNet. An easy way out would be to use a POS tagger to identify and filter out proper nouns.

Another issue is that we have not used any compound nouns (*e.g.*, police officer, dinning table, printer cat ridge, *etc.*) as we did not have access to a shallow parser. Intuitively, identifying and using compound nouns to compute the chains will give better results than just using the individual nouns. Even in the above case, chains 3 and 4 should have ideally been a single chain referring

to a 'station wagon - a car that has a long body and rear door with space behind rear seat' which is the actual sense in the document as opposed to a 'a wheeled vehicle drawn by a horse or tractor'!

The chain evaluation heuristic used is a very simple technique based on the thresholded normalised chain length. Even with this simple measure we are able to select good lexical chains. Nevertheless, we intend to investigate the more complex chain evaluation methods that have been proposed in the literature for the other applications of lexical chains.

5 Concluding Remarks

The performance of the clustering algorithm is closely tied to the quality of the lexical chains used to represent the documents. Some issues such as proper nouns, compound nouns, word sense disambiguation and overall chain quality when solved will help improve the clustering quality. Instead of just removing the proper nouns from the input, we intend to work on resolving the proper noun problem. If properly resolved, we feel that proper nouns together with compound nouns will help us significantly to improve the chain quality and hence the clustering quality. Word sense disambiguation (WSD) is an actively researched topic in the context of WordNet and various proposals exist for doing it. We intend to critically evaluate these proposals and see how we can enhance the WSD capability in our algorithm.

One other enhancement that we plan to explore is to try to identify and group the lexical chains formed based on the hypernymy relations. For example, our algorithm would generate separate lexical chains as follows for the words *car, auto* and *vehicle*:

- car, auto, motor car
- truck, motor truck
- motor vehicle

The first two groups are related to the third group by a hypernym relation in WordNet. This we expect would result in a hierarchy of clusters, with clusters reducing in granularity as we go up the hierarchy. We believe that identifying and marking such semantic relations between the clusters will result in a more semantically sensible clustering of the document.

Almost all existing algorithms for computing lexical chains seem to use only noun words, ignoring other parts of speech. We intend to evaluate effectiveness of using verbs and other parts of speech as well, in computing the lexical chains.

We have run our experiments on a small dataset of 30 documents. This was because we were unable to get a pre-clustered dataset and comparing the results would have been difficult. Hence, we limited our experiments to a small dataset to keep our experiments humanly tractable. Our algorithm is easily scalable to a larger number of documents. In the future, we plan to evaluate it on the significant portion of the 20 Newsgroups and RCV1 corpus. We conclude with the comment that our method can be used to create a hierarchy of document clusters and can be effectively extended for automatic topic detection.

6 Acknowledgement

The authors would like to thank Director, CAIR for the encouragement and support given for this work

References

[1] Xu, R., II, D.W.: Survey of clustering algorithms. IEEE Transactions on Neural Networks **16** (2005) 645– 678
[2] Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18** (1975) 613–620
[3] Halliday, M.A.K., Hasan, R.: Cohesion in English. Longman (1976)
[4] Barzilay, R., Elhadad, M.: Using lexical chains for text summarization. In: In Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL, Madrid, Spain. (1997)
[5] Silber, H.G., McCoy, K.F.: Efficiently computed lexical chains as an intermediate representation for automatic text summarization. Computational Linguistics **28** (2002) 487–496
[6] Hirst, G., St-Onge, D.: Lexical chains as representation of context for the detection and correction malapropisms. In Fellbaum, C., ed.: WordNet:An electronic lexical database and some of its applications. The MIT Press, Cambrige, MA. (1997)
[7] Green, S.J.: Automatically generating hypertext in newspaper articles by computing semantic relatedness. In Powers, D., ed.: NeMLaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language. (1998)
[8] Morris, J., Hirst, G.: Lexical cohesion computed by thesaural relations as an indicator of the structure of text. Computational Linguistics **17** (1991) 21–48
[9] Fellbaum, C., ed.: WordNet: An Electronic Lexical Database. MIT Press (1998)
[10] Jarmasz, M., Szpakowicz, S.: 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. (2003)
[11] Kirkpatrick, B.: Roget's Thesaurus of English Words and Phrases. Penguin (1998)
[12] Jarmasz, M., Szpakowicz, S.: The design and implementation of an electronic lexical knowledge base. In: Proceedings of the 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence (AI 2001), Ottawa, Canada. (2001) 325–334
[13] Patwardhan, S., Banerjee, S., Pedersen, T.: 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. (2003)
[14] Pedersen, T., Banerjee, S., Patwardhan, S., Michelizzi, J.: SenseRelate Project (2004)
[15] : 20 Newsgroups dataset. (Online: http://people.csail.mit.edu/jrennie/20Newsgroups/)
[16] Frakes, W.B., Baeza-Yates, R., eds.: Information retrieval: data structures and algorithms. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1992)
[17] Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: KDD Workshop on Text Mining. (2000)
[18] Karypis, G.: Cluto - a clustering toolkit. Technical Report 02-017, University of Minnesota - Computer Science and Engineering (2002)

[19] Zamir, O., Etzioni, O.: Web document clustering: a feasibility demonstration. In: SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (1998) 46–54