Using WordNet Similarity and Antonymy Relations to Aid Document Retrieval

Thomas de Simone and **Dimitar Kazakov**
Department of Computer Science
University of York
Heslington, York, YO10 5DD, UK
mrtom@tomandpete.co.uk, kazakov@cs.york.ac.uk

Abstract

This article proposes a document search technique that uses the language database WordNet to cluster search results into meaningful categories according to the words that modify the original search term in the text. The work focusses on the relevance of the semantic relations of similarity and antonymy as present in WordNet.

1 Introduction

The growing issue of information overload on the Internet has prompted the appearance of a number of clustered search strategies which all have the same goal of breaking down large sets of search results into smaller clusters, each cluster containing results which should be equally relevant to each other. The most popular current techniques found on the Internet follow an online clustering process whereby the user inputs a search term, and the returned documents are compared to each other to decide which ones belong together. It is this method of relating documents to each other which is under heavy investigation, with many different approaches being suggested. Clustering search engines[1] tend to use string matching, rather than true linguistic analysis, to identify key words and phrases which documents share in common, and generate clusters based on these phrases. In this work, the words are semantically analysed using WordNet (WN) (Miller *et al.*, 1993) to give a more informed analysis of how the documents might relate to each other. The WN relations studied are those of similarity and antonymy. If the search word is a noun or verb, the matching documents could be clustered according to the adjectives, resp. adverbs modifying the search term in the text.

In addition to WN, the work also makes use of PoS tagging, shallow parsing, Tgrep (Pito, 1992) and entropy (Shannon, 1948). The primary corpus used for the study is the Wall Street Journal (WSJ) treebank, but the Semcor corpus[2] is also used to investigate the potential benefits offered by Word Sense Disambiguation (WSD). Semcor is a text corpus which has been manually annotated with WN sense tags, giving the effect of being processed with a high quality WSD algorithm.

2 Related Work

Zamir and Etzioni's on-line clustering tool *Grouper* (Zamir and Etzioni, 1999) is representative of the current academic research using phrase matching. The tool uses a linear time agglomerative clustering algorithm known as Suffix Tree Clustering (STC). It begins by identifying the most frequent phrases, and then clustering together documents which share them, forming base clusters. Clusters are allowed to overlap, and can be merged if the overlap is considerable. Clusters are then named according to its most representative phrase. The latter is labelled with its relative frequency in the cluster.

Investigations in the use of WN in document clustering include work by Hotho *et al.* (2003), and Sedding and Kazakov (2004). Both works focus on off-line clustering, and preprocess documents using WN as follows (with some variations between the two). Each document is PoS tagged, stemmed, and converted into a "bag of words" representation, the result being that each document is represented as a list of words and their frequencies within the document. Words are then replaced with tokens representing their synonyms and hypernyms. This means that two words which are synonymous would now be represented by the same token. In the case of hypernyms, two words are represented by the same token when one is a more general concept subsuming the other. The frequencies of all features are then modified using the `tf idf` weighting scheme in order to highlight features with high

[1] See http://www.clusty.com, http://www.dogpile.com and Grouper (Zamir and Etzioni, 1999), among others.

[2] The Semcor files are freely available to download from http://www.cs.unt.edu/~rada/downloads.html

frequency in selected documents, but infrequent on the whole (Salton and Wong, 1975). A clustering algorithm (bisectional k-means) is then applied. The results indicate that while using the synonymy relation does indeed have a beneficial effect on the quality of clusters, the inclusion of hypernyms produces acceptable results only when carefully managed, as the `tf idf` weighting on its own is not sufficient to eliminate the negative influence of very general hypernyms.

3 Clustering with Synonymy and Antonymy

This work echoes some of the ideas presented in the previous section, but studies a different aspect of the possible contribution of WN to clustering, and makes no use of the hypernym relation. The main idea is as follows: a collection of documents containing a given noun or verb can be subdivided according to the modifiers (adjectives, resp. adverbs) of that search term in the set of documents. Documents containing the same adjective (resp. adverb) can be grouped in the same cluster. The resulting clusters can also be aggregated by bringing together modifiers linked through the WN similarity relation. In either case, the user could be automatically provided with meaningful alternatives (in the shape of pairs of opposite concepts) to further refine the search if one or more pairs of clusters formed in the above way corresponded to pairs of WN antonyms. Ideally, such pairs of clusters would cover a large proportion of documents, and be of equal (or similar) size.

If, in a given search, two such clusters can be found which are sufficiently large to cover the majority of the search results, then subsequent subclusters could be derived from them.

Given that there will invariably be documents which still do not belong to a cluster after this process, it is also investigated how the remaining results could be clustered. In this case, singleton clusters could also be produced based on modifiers which are linked together via the similarity relation. Again, it is hoped that the resulting clusters would be small in number, and between them would cover the majority of the set of search results.

4 Design

The tool is split into two separate functions: initial preprocessing and on-line searching. The first stage uses morpho-lexical analysis, PoS tagging and shallow parsing to replace word forms by their standard lexical entries, and reduce documents from a coherent collection of text to a "bag of features" representation, where each feature in a document contains a head and a corresponding modifier. Optionally, lexical entries can be replaced by their synsets for additional aggregation of features. In case of semantic ambiguity, only the most frequent sense/synset (as listed in WN) is used for the WSJ as a crude form of word sense disambiguation (WSD). In the case of Semcor, the correct synset listed in the corpus is used to emulate the performance of a perfect WSD tool.

The searchable database is represented as a hierarchical set of records, as depicted in Figure 1. The database is split into two "docspaces", which are collections of documents preprocessed for search with nouns, resp. verbs as keywords. For each docspace and document, there is a set of records representing each noun (verb) in that document, and the modifiers with which it is linked in the text. These records also store the frequencies with which the head word and each of its modifiers appear in the document.

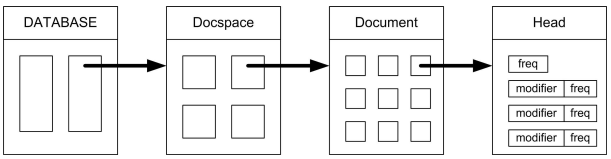

Figure 1: DB structure after preprocessing.

At search time, the keyword is preprocessed in the same way as the documents in the database (to ensure consistent treatment of words), and then all documents containing this keyword are retrieved. Each document is linked to the list of keyword (phrase head) modifiers it contains. An example of this is given in Figure 2. Note that at this point, it is no longer necessary to refer to the actual head, because all of these modifiers are describing the same head.

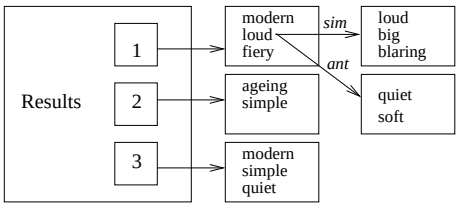

Figure 2: Representation of initial results.

Here the similarity relation can be used, in two separate ways. On its own, it allows to expand each modifier into a list (or adjective/adverb "sim" cluster), containing the original modifier and all the words (adjectives or adverbs) it is similar to. In combination with antonymy, similarity can produce the list of indirect antonyms to the modifier in question. Figure 2 shows how "loud" is expanded into two clusters, one representing adjectives with similar meaning, and another representing the opposite.

In all cases, the ultimate goal is to attempt splitting the documents into pairs of subsets, containing the search keyword with a modifier or its antonym. If neither similarity nor antonymy has been used in the preceding steps, the algorithm will explore only the direct antonymy relation to create these pairs of subsets. If similarity and antonymy are used in the previous step, it is also possible to generate pairs of clusters representing indirect antonyms.

The actual algorithm takes a document's list of modifiers and looks for occurrences of each modifier (and its synonyms and antonyms, if applicable) in the other documents. Given the example in Figure 2, the algorithm would start with the modifiers from document 1 and search for them within documents 2 and 3. It would then search for all the modifiers from document 2 in document 3 (but will not need to look back in document 1). In the illustrated example, documents 1 and 3 will seed a pair of antonym ("ant") clusters defined by "loud" and "quiet".

Entropy (Shannon, 1948) has been used to score the quality of a pair of ant clusters, or of a sim cluster, together with its complement. (In fact, a good case can also be made for the use of information gain (Quinlan, 1986)). In an interactive mode, this score helps list the best pairs of antonyms first, and could potentially be used when hierarchical clustering is considered. However, as the results on the coverage of antonym clusters in the next section suggest, hierarchical clustering is possible only in a small fraction of cases.

For two antonyms a and b, we can define the set of documents covered by them as A_1 and A_2 respectively. We can also define the set of all documents in the current search as the universe, U. The most useful pair of antonyms a and b will be that which comes closest to satisfying the following equations: $A_1 \cup A_2 = U$; $A_1 \cap A_2 = \emptyset$. We then define $m = |A_1 \setminus \{A_1 \cap A_2\}|$, and $n = |A_2 \setminus \{A_1 \cap A_2\}|$. Here m is the number of documents containing modifier a, but not modifier b (and vice versa for n). These can be used to express the entropy equation, as shown below.

$$Ent(S) = -\sum_{i \in \{m,\, n\}} \left(\frac{i}{m+n}\right) \log_2 \left(\frac{i}{m+n}\right)$$

The best pair of antonym clusters will be the one with maximum entropy.

5 Implementation

The main program is written in Perl and interacts with the WN database via the third party package `WordNet::QueryData`[3]. The preprocessing stages are largely taken care of by a subset of the WSJ treebank that has been fully parsed and PoS tagged, presented in a format searchable by Tgrep. This allows features to be extracted using Tgrep's powerful search language, which can search parse trees and use PoS tags and regular expressions to find features. For instance, all adjective-noun pairs in noun phrases ending in "`Adjs N`" can be extracted, e.g., "The short, wealthy, Irish director" → (`short director`), (`wealthy director`), etc. These features are written to a text file, which can then be parsed by the program and turned into a database structure, as illustrated in the previous section. When presented with PoS tags attached, these features can also be preprocessed by WN.

The search tool uses another third party package `Set::Scalar`[4] to convert the arrays of documents produced in a search into set representations, thus allowing the equations from the previous section to be used with ease in the program.

6 Results

Both WSJ and Semcor were converted into "bag of features" representations under a range of configurations. The noun-based docspace representing the WSJ treebank was used to produce the following four data sets:

wsj_jjnn Only stemming is performed on the corpus; no words are grouped into synsets.

[3] Available on `www.cpan.org`.

[4] Also available on `www.cpan.org`.

wsj_jjnn_psyn Both heads and modifiers are replaced with the synset ID representing the most common (primary) sense of the word.

wsj_jjnn_psh Only heads are replaced with primary synset IDs.

wsj_jjnn_psm Only modifiers are replaced with primary synset IDs.

Each of these configurations was batch tested using the complete list of possible search terms (nouns) present in the data. The clustering algorithm was run twice with the following options:

Sim, ant Similarity and indirect antonyms used for clustering.

No sim, no ant Neither of the above used.

In addition, a data set comprising nouns modified by other nouns, and a verb-based data set, were created and batch tested using the "No sim, no ant" strategy, since they were only needed to indicate rough figures for the potential inclusion of these two types of feature in the search tool. They were called **wsj_nnnn** and **wsj_vbrb** respectively.

The Semcor corpus was tested in two configurations for both nouns and verbs:

sem_1jjnn Noun-related data

sem_1jjnn_wsd Noun-related data with sense-tagged words

sem_vbrb Verb-related data

sem_vbrb_wsd Verb-related data with sense-tagged words.

Key results are presented below, with a full set of results available from the second author's Web publication list.[5]

The "Occurrence of Antonyms" test investigates how many results are found containing one or more antonym pairs over the set of search terms. The original hypothesis suggests that the majority of results will contain at least one antonym pair which can then be further broken down. Figure 3 shows the experimental results gathered for the key configurations. With the first three configurations, the total number of possible search terms is in the region of 2000, but when using sense tags with the Semcor corpus, this number rises to 2500.

[5] www.cs.york.ac.uk/~kazakov/my-publications.html

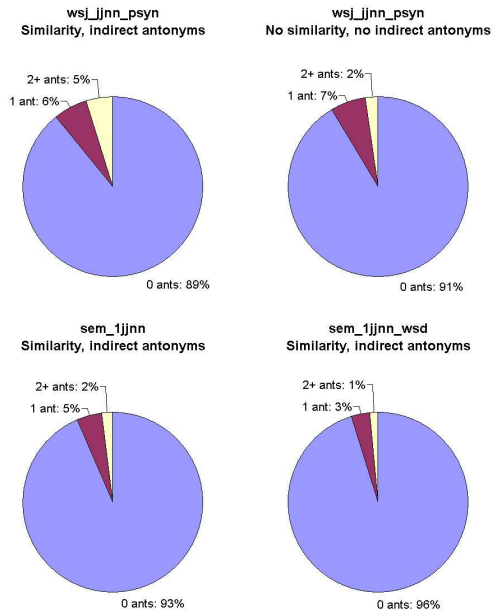

Figure 3: "Occurrence of Antonyms" results for WSJ.

These results show that over all possible searches, the vast majority (typically in the region of 90%) return results that contain no antonym pairs at all. Of the remaining 10%, roughly half contain only a single antonym pair. In some test cases, searches are found to produce as many as 12 pairs, but in most cases, there are rarely more than a handful of search terms in a set which produce more than four pairs. This means that there is simply not a great enough proportion of searches which return even one antonym pair to split further. This evidence is backed up by the results from Semcor, which show a very similar trend over a different corpus.

In terms of entropy and coverage, the average figures for ant pairs in **wsj_jjnn_psyn** with similarity enabled is 0.51, which is quite far from the ideal value of 1. This means that the separation seen in most ant pairs tends to be uneven. The average coverage of ant pairs (the proportion of documents containing the search term and a modifier from either antonym cluster) in the same configuration is just 15%, meaning that the pairs do not actually account for a very substantial portion of the search results. The figures are quoted for this configuration because it is the configuration which produced the best results.

It can also be seen from the first two pie charts in Figure 3 that the use of the similarity relation changes the proportions of results con-

Table 1: Impact of similarity on clusters.

	sim & ind ant	neither
Avg. no. sims	9.86	10.58
Avg. coverage	11%	6%

taining antonyms. When similarity and indirect antonyms are employed, the number of results containing no antonym pairs decreases, if only by a small amount. This means that the similarity relation causes more clusters to link via the antonymy relation, as would be expected.

The impact of the similarity relation is further highlighted in Table 1, where it can be seen that its use with **wsj_jjnn** causes the number of sim clusters to drop, and the average coverage of each cluster (its size) to increase. Again, this is in line with the expected result, since more clusters are being linked together, and joining to form larger clusters.

As for the use of verb-adverb features, the **wsj_vbrb** configuration proved to be of much more limited use. Semcor produced a grand total of 348 possible search terms, compared to the noun-adjective features, which produce in the region of 2000 search terms. For WSJ, the number of verb-adverb features was much lower, to the point of being useless.

The **wsj_nnnn** configuration (noun-noun features) produced 1257 possible search terms, which may be smaller than the number produced by the data sets for adjective-noun features, but is large enough to be significant. The search for singleton clusters (analogue to the sim clusters) based on an exact match of a noun modifier produced the results in Table 2.

7 Conclusions

The primary goal was to establish if the antonymy relation could be used on the modifiers found in documents to decompose a set of search results into a hierarchy of sub-clusters. The experiments performed suggest, with a convincing majority, that this is not possible. In all experiments, a small proportion of search terms produced results containing one antonym pair, and even then the values for entropy and coverage were not as high as was hoped for. This means that even for those search terms which do return antonym pairs, the quality of those pairs is far from ideal. The use of indirect antonyms did indeed increase the average number of antonym pairs found, as would be expected, but this was still not enough to use this clustering technique in isolation. This work has also shown that compared to nouns, verbs offer a limited amount of help for document clustering.

Table 2: Sim clusters of NN phrases.

	wsj_nnnn
Total search terms	1257
Average number of clusters	5.54
Average coverage	0.42

An analysis of the make-up of selected search results shows that the average search result can be expected to contain at most one or two large clusters, and the remainder of the results tend to be a large collection of very small clusters, typically only one or two documents in size. This is very much counter to the hypothesis expressed earlier in the report, and suggests that if modifiers are going to be used as the features which make up clusters, some sort of agglomerative approach needs to be implemented after the primary clusters are returned, in order to turn these insignificant clusters into a smaller collection of larger clusters. One possible idea would be to group clusters containing the same (or similar) modifier and heads linked through hypernymy (Hotho *et al.*, 2003; Kazakov and Sedding, 2004). In this way, the two approaches might offset each other's drawbacks somewhat.

Finally, one should seriously consider using the sim clusters of noun-noun `NP` phrases, the coverage of which appears nearly optimal.

References

A. Hotho, S. Staab, and G. Stumme. Wordnet Improves Text Document Clustering, 2003.

D. Kazakov and J. Sedding. Wordnet-Based Text Document Clustering. *Third Workshop on Robust Methods in Analysis of Natural Language Data (ROMAND)*, pages 104–113, 2004.

G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five Papers on Wordnet. *Tech. report, Princeton University*, 1993.

R. Pito. Tgrep manual. Distributed with the Penn Treebank, 1992.

J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

G. Salton and A. Wong. A vector space model for automatic indexing. *Communications of the ACM*, 18:613–620, 1975.

C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, 1948.

O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks*, 31(11–16):1361–1374, 1999.