Automatic Acquisition of Word Classification Using Distribution Analysis of Content Words with Respect to Function Words

Andrew Roberts

Computer Science
(2001/2002)

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) _____

Summary

This project describes a method which can automatically infer word classification. Previous systems designed to assign parts-of-speech to words sought the use of training data or were built upon rules devised by experts in linguistics. The report details the use of an *unsupervised* approach that can reduce significantly the reliance on prior linguistic intuition.

The study looks in to how words behave relative to the function words. As these are the most common words, there is a great deal of information that can be attained. It was possible to analyse how the content words from a given body of text were distributed with respect to the function words. This information could be used as a profile, and therefore content words with a similar profile against the function words could be assumed to be of similar word class.

Agglomerative hierarchical clustering techniques were applied to partition words into different clusters. Words that were deemed similar were grouped together, and thus, each cluster should contain words that posses the same part-of-speech.

This project performed many experiments to investigate how the many factors affected the overall clustering performance, in order to find the optimal parameters. The results report an accuracy of 87% when performed on the LOB corpus. Experiments were also carried out with an alternative Spanish corpus and the clustering accuracy achieved 85%. Semantic clustering was also observed indicating the effectiveness of the described approach for the task of automatically acquiring word classification.

Acknowledgements

I would like to thank my project supervisor, Clive Souter for his guidance, advice, patience and for the cups of tea.

Thanks to Bill Whyte and John Elliott who conceived the idea for the project (see appendix B). Also, to their kindness for allowing me to frequently interrupt their time to discuss ideas and problems. Thanks also to Eric Atwell for his inspiration and support.

I am grateful to Alex Boag-Munroe for kindly allowing me to remotely log in and hog his super-fast PC so that I could perform a large proportion of the experiments.

Finally, thanks to my family and friends for coping admirably with my moods and stress during the time spent working on this project, notably, Alex Morrison, Chris Walton and Ron Hamiliton.

Especial thanks to Leonora.

Contents

1 Introduction

1.1 What's the problem?

This project primarily deals with attempting to differentiate words according to their word classification, e.g., is it a noun? Or a verb? Etc. This process is referred to as tagging, and is a task found within the field of Natural Language Processing (NLP).

Tagging is described in more detail in section 2. It is important to appreciate, however, that tagging has an important role within NLP and helps to provide a solid foundation for many larger applications which require the information that tagging can provide. It is for that reason that taggers have been around as early as the 1960's.

Early taggers relied on expert linguists to define rules which would then be able to assign a word with its respective classification. The next breed of taggers used statistical techniques to calculate probabilities in order to determine the word classification. This still requires experts in order to gain maximum potential out of such a tagger, as it needs to be trained in order to be effective.

The aim of this project has been to attempt a more novel approach than those used in taggers over the past 40 years. In more recent years, the idea of utilising unsupervised techniques in tagging has showed promising results. This essentially involves looking at how words behave relative to each other. By analysing which words appear in close proximity (and for that matter, also those that do not) it is possible to gain a great deal of information that can be used to partition words into their correct classification, with the need of prior expert knowledge.

More specifically, looking at how words behave relative to function words is what this research concentrates on. Function words are a small set of words which occur most frequently in the English language and play an important part in grammar. It is therefore reasonable to expect that they possess lots of information that can be extracted to acquire word classification. Investigating that hypothesis is the purpose of this project.

1.2 Aims and objectives

1.2.1 Overall Aim Of The Project

The aim of the project is to apply Natural Language Learning (NLL) techniques to develop methods to determine the classification of words automatically.

1.2.2 Objectives Of The Project

1. To investigate the current techniques used to tag words into parts-of-speech.
2. Look at NLL techniques to determine whether any may be useful for solving the problem.
3. Obtain a variety of samples of written text on which to apply NLL techniques on.
4. Develop an algorithm to automatically classify words from an inputted sample of text.
5. Evaluate and verify accuracy of algorithm by comparing results to tagged corpora created by expert linguists.

1.2.3 Minimum Requirements Of The Project

1. To produce an algorithm that can analyse the relationship between content words and a given set of function words.
2. By looking at such relationships, extend the algorithm to cluster words which behave similarly to each other, relative to the function words.
3. Verify that the majority of content words in each cluster are of an equivalent class of words, e.g., nouns.
4. To test the algorithm on English plus one other language.

1.3 Scope

Whilst the majority of this report focuses on what the project has done, it is worth noting now what this project has not done. This is because with work of this nature, there is no limit to the depth of research which could potentially be carried out. The purpose of this sub-section is therefore to outline the main boundaries to define the scope of this project.

The objective of this project is not to produce a highly accurate tagger per se; it is to develop a method which requires as little linguistic intuition as possible to acquire word classification, and investigate the effectiveness of that method by experimenting and evaluating its accuracy. If it transpired that the method implemented was only 10% accurate, then the aims and objectives of the project would still have been met – with the conclusion that the proposed method is unsuitable for this task.

Another worthwhile point is that this project is *not* a software engineering exercise. Although a considerable amount of programming is involved to implement the solution – the software is the tool for evaluating the feasibility of the proposed method to automatically acquire word classification. Therefore, despite belief of the author that the implementation is efficient, and more importantly, correct, there are no guarantees that it is so. Design and testing for the coding of the software will not feature in this report.

1.4 Structure of report

Section 2 introduces all the relevant background information that aims to provide a broad understanding of all topics and principles that are covered throughout the rest of the report. Section 3 will discuss the approach that was used to develop a method of automatically classifying words. Section 4 goes on to describe the implementation of the algorithms outlined in section 3 and how they are used to obtain the results. Section 5 summarises the experiments performed using the software to discover what influence the many factors have on the overall clustering accuracy. Section 6 examines potential work for the future. Finally, section 7 provides the conclusions of the project, reporting on what has been achieved and experimental findings.

2 Background

2.1 Word Classes

The English language can be broken down into parts-of-speech. At the highest level, there are closed class types and open class types. Closed classes are so called because they remain relatively permanent. The closed classes all tend to be function words, therefore, examples would be pronouns, prepositions, conjunctions etc. Such classes remain comparatively static, in that new function words rarely occur. In contrast, the open classes do not remain fixed; examples of open classes are nouns and verbs, of which new words of these types are added continuously (Jurafsky and Martin 2000). The rate at which new words are "discovered" is considerable. The Oxford English Dictionary (Simpson and Weiner 1989) – seen by many as the authority for its coverage of the English language, past and present – has a team of skilled linguists whose job is to "discover" new words and gather evidence of its usage. Subscribers to the OED online service can expect the benefit of "at least 1,000 new and revised words will be released each quarter". Such a statement illustrates not only the rate of which new words are adopted in everyday language, but also to how defined words are adapted to provide new meanings.

N.B. Word class and part-of-speech will be used interchangeably in this report.

2.2 Corpora

A corpus is essentially a collection of text. In order to expand, corpus samples are often taken from newspaper articles, novels etc. Popular English corpus resources include:

- Brown Corpus (Kucera and Francis 1967, Francis 1979 and Francis and Kucera 1982)
- LOB Corpus (Johansson et al. 1986)
- COBUILD Corpus (Sinclair 1987)
- British National Corpus (Leech 1993)

Each corpus contains various quantities of words, from different samples. For example, the Brown corpus was designed as a representative sample of written American English. The LOB corpus was created as a British English equivalent.

Corpora are not restricted to collections of written text. There exist many spoken word corpora. Such a corpus is a collection of spoken utterances. It is worth noting that the words, and their usages can vary tremendously – it should not come as any surprise to reveal that the language used when a person speaks will often be different to the language they use when writing. Examples of spoken corpora:

- London Lund Corpus (Svartvik 1990)
- Lancaster/IBM Spoken English Corpus (SEC) (Taylor and Knowles 1988)
- Corpus of London Teenage Language (COLT) (Haslerud and Stenström 1995)

2.3 Tagging

The discipline of Natural Language Processing has long sought to investigate the information that can be gained by analysing how words of different parts-of-speech behave relative to each other (Manning and Schütze 2000). In order to do that, there must exist a large corpus of words, each respectively assigned a *tag* to indicate which word class they belong to.

To achieve a tagged corpus, a *tagset* must first be defined. Tagsets can vary enormously in size from 40 to 200 tags (Jurafsky and Martin 2000). Of course, a significant number of those tags are associated with punctuation. See appendix C for the tagset used by the LOB corpus. Popular tagsets include:

- 45-tag Penn Treebank tagset (Marcus *et al*. 1993)
- 61 tag C5 tagset used for the British National Corpus (Garside *et al*. 1997)
- 87-tag tagset used for the Brown corpus (Francis 1979; Francis and Kucera 1989)
- 146-tag C7 tagset (Leech *et al*. 1994)

It would be unfeasible to expect an expert linguist to manually tag the words of an entire corpus at is could contain hundreds of thousands, if not millions of words! Instead, automatic algorithms to perform this task have been created; these fall into two main categories: rule-based tagging and stochastic. Rule-based tagging – as the name suggests – relies upon hand-written rules which define constraints to ensure correctness. A typical approach uses a two-phase approach. See early examples of this approach in Harris (1962); Klein and Simmons (1963); Greene and Rubin (1971). The first stage uses a dictionary lookup containing words and their potential parts-of-speech. The second stage is to then

apply the rules to remove any ambiguity, thus reducing the list to a single tag for each word. Stochastic tagging involves the use of probabilities to determine the most likely tag for a word, such as HMM[1]-based or cue-based. Using probabilities in tagging is an obvious approach and has been utilised for many decades. First used by Stolz *et al*. (1965), various stochastic taggers include Marshall (1983), Garside (1987), and Church (1988). HMM taggers have to be trained on previously tagged data in order to calculate the probabilities for tag sequences. Also, it has been demonstrated that with the use of the Expectation-Maximisation (EM) algorithm (Dempster *et al*. 1977), that stochastic models can be trained on untagged data (Cutting *et al*. 1992). However, this method still requires a dictionary of words with their respective word tags. The EM algorithm is then able to calculate the likelihood for each tag and tag-transition probabilities. Experiments so far, however, have shown that taggers trained on tagged data will perform better than ones employing the EM algorithm (Merialdo 1994).

2.4 Ambiguity

The majority of words can easily be classified because they only belong to a single word class. However, many of the common words used can have more than one part-of-speech. For example, 'dish' can have more than one usage: as a noun in *dish of soup*, or as a verb, as in *dish the soup*. Such cases are known as ambiguous. With respect to tagging, it is up to the algorithm employed to disambiguate words by ensuring they take the context of its usage into consideration.

2.5 Function Words

For this project, particular attention will be focused on the function words that appear within a given text. As mentioned previously, function words tend to fall into the closed classes. The closed classes are (with examples which are by no means exhaustive):

- Prepositions: on, at, to, of
- Determiners: a, an, the
- Pronouns: she, who, I
- Conjunctions: and, but, if, or

[1] Hidden Markov Model

- Auxiliary verbs: can are, may
- Particles: up, in, by

Function words have an important part in grammar. They appear the most frequently, which also explains why they are short in length. In the development of language, common words are also short words because they require less effort to use. (Zipf, 1949)

2.6 Automatically Acquiring Word Classification

This project will focus primarily on the automatic acquisition of word classification. The difference between this method and other part-of-speech tagging essentially comes down the amount of expert linguistic knowledge the algorithm has incorporated. Tagging algorithms rely on prior knowledge of the language such as grammatical rules and precompiled lists of words with their possible tags. Automatically determining parts-of-speech on the other-hand presumes no prior intuition and should be ignorant of the given language, lexical types and grammatical rules – or, it is often referred to as *unsupervised*.

A popular and straightforward approach to solve the problem is by means of *distributional analysis*. A statistical method of looking at where word types are positioned most frequently within a sentence.

Its usefulness has been called into question. Chomsky has never been enthusiastic about distributional linguistics (Chomsky 1964), or the automatic discovery of grammars, concluding: “I think it is very questionable that this goal is attainable in any interesting way” (Chomsky 1957). It is his belief that linguistic knowledge is already formed in the brain and not acquired during learning because language is sparse and variable. However, the debate to the innateness of language in humans need not concern this project.

Data sparseness is a problem throughout the domain of NLP. This can be explained using Zipf’s Law. He observed (Zipf 1949) that if you list words in order of their frequency of occurrence, then the relationship between the frequency of a word *f* and its position in the list, known as its rank *r*, can be described as:

$$f \propto \frac{1}{r}$$

The implication of this law is that the vast majority of words that appear in a given corpus will be sparse, and for relatively few common words will there be many examples (Manning and Schütze 2000). For words that do not occur frequently, it is difficult to reliably deduce information about them. One way to combat the problem is to simply use a larger corpus of words, thus increasing the probability of rarer words to appear. Unfortunately, this is often infeasible as the computational demands of processing large numbers of words becomes astronomical (this is less restricting due to the ever increasing performance of computers). Additionally, the greater the size of corpus, new words with sparse distributions appear. Another possible approach is the hand-pick special examples to ensure certain words, word combinations etc., are analysed. However, the overall result will not be representational of common word usage as found in corpora.

2.7 Relevant Research

Kiss (1972), in the context of understanding the psychology of learning, studied the order of acquisition of words in children. He showed that it was statistically possible to distinguish different word classes based on their distributional properties. His approach took 30 words from language found in children's literature and clustered them based on their similarity to their nearest neighbour bigram[2] counts. He was restricted to only being able to classify few words.

Baker (1975, 1979) published a model which originally intended as a procedure for automatic training of speech recognition systems. However, it happened that it could be also used for the purpose of tagging parts-of-speech. Assuming that a language could be generated by a Markov process, then he proposed a technique to automatically calculate the parameters of a Markov model which was compatible with the data. Unfortunately, Baker's recursive formulae for estimating the parameters of a Markov model was computationally expensive which prevented any practical application (Atwell 1987).

Atwell (1987) attempted to apply the theory put forward by Baker (1975, 1979). He had to place considerable constraints in order to make the computational demands more reasonable; for example, the assumption that a word can only belong to a single word class. Also, he only considered words located immediately before and after each designated word

[2] An *n*-gram is a sequence of *n* successive objects. In this instance, the objects are words, and *bi* represents $n = 2$.

(although this was performed separately because processing was still very slow). Despite some promising outcomes, Atwell acknowledged that his sample was too small to provide conclusive results (at the time, using a sample of 200,000 words, the program took several weeks on a mainframe computer to complete the task), and that the constraint on single word class participation would need to be removed for the system to be wholly successful.

Hughes (1994) performed many experiments in order to find which factors give the best accuracy when automatically determining parts-of-speech. The variables were:

1. The contextual pattern, e.g., sentence position distribution; various sized bigrams.
2. The metric used to calculate the distance between words in vector space.
3. The clustering method, e.g., Single linkage; Centre of gravity; Ward's method etc.
4. The size of the comparison set.

With a 35 million word corpus available, generated from extracting USENET articles, the 200 most frequent words were used in the experiments, which were adequate to evaluate the most effective factors. Once the best combination had been established, he clustered the top 2000 occurring words from his corpus with a high degree of success: 87% accuracy if the words are classified into 100 clusters.

Hughes found that the distributional redundancy of word types was revealed using bigrams rather than absolute sentence position. And that the context of absolute sentence position distribution was contained within the context of bigram counts.

Redington *et al*. (1998) undertook research into automatically acquiring syntactic categories in the psychological context of attempting to understand the processes of how children learn language showed good success. A particularly relevant experiment performed was clustering a set of words with the function words removed. The reasoning for the test was that child speech is sparse in function words in comparison the proportion of content words they use. If they are effectively only concentrating on content words, then a child's mechanism for acquiring word classes does not require function words. That specific experiment did show that word classification was in fact possible when function words were excluded from the input stream. However, they did acknowledge that removal of the function words did have a "considerable impact" on the informativeness of the results.

2.8 Clustering

The principle of clustering algorithms is to divide a set of objects in to clusters. By using a given measure of similarity, a good clustering algorithm will place objects that are similar into the same cluster, whereas dissimilar ones are clustered into different groups.

Clustering is ideal for the automatic acquisition of parts-of-speech because is *unsupervised* – in that it doesn't require training of class labels to partition groups.

Clustering algorithms fall broadly into a few fundamental types. The first relates to the structure of clustering that is employed. *Flat clustering* is simple in structure: consisting of a particular number of clusters, where the relations between them are often undetermined. The alternative is *hierarchical clustering*. This can be represented as a tree structure, where the root is the entire set of objects, and each node corresponds to a subclass of its parent node. The leaves of the tree are the individual objects that have been clustered. Another type of clustering is defined by membership of objects in clusters. *Hard clustering* has the constraint that an object can only belong to a single cluster. If membership in multiple clusters is permitted, then the clustering algorithm has performed soft clustering (Manning and Schütze 2000). Hard clustering in the context of this project has obvious implications with regard to ambiguity of word types, and will need to be addressed during the selection of the appropriate algorithm to solve the task.

The hierarchical method is the most applicable to the problem at hand. The clustering methods used in this project are discussed in section 3.3

There is clearly a trade-off in choosing the right algorithm between the performance of the clustering (with respect to the accuracy of the results) and its computation demands. Clustering is a very expensive process, therefore efficiency is very important to allow a large enough number of words to be clustered for the results to be conclusive. Previous research, as mentioned above, have all suffered because of the restrictions caused by implausible computing time of large samples. It is difficult to tell which particular method will give the best results. It will therefore be necessary to experiment with various algorithms to establish the one which gives optimal performance.

2.9 Summary

Applying Natural Language Learning techniques to automatically determine parts-of-speech is still in its infancy. This is due to progress being hindered by the huge computational demands the problem exhibits. However, the results from the research discussed earlier are promising enough to boost confidence that attempting to automatically acquire word classes is not only feasible, but can be highly accurate.

By only concentrating on the role of the function words, and how content words are distributed in relation to them - as opposed to looking at how each word interacts with every other word - this should reduce the intensity of processing required to cluster the content words into their respective word classes. Of course, there is the risk that the results may not be as accurate due to that very fact that only the relationship between function words and content words is being analysed. The experiment carried out by Redington *et al*. (1998) showed that by not including the function words in the analysis, results were drastically affected, which bodes well in the assumption that due to the importance of function words in grammar, they are rich in syntactic information. It is ultimately the purpose of this project to evaluate the effectiveness of this approach.

The overall goal of this line of research is worthwhile. Not only does it offer attractive prospects for the use in deciphering unknown languages. As Atwell and Drakos (1987) concluded, the 'bottleneck' in commercial exploitation of NLP systems would be resolved because tailoring specialised applications could be automated in order to provide solutions for a wide range of applications.

3 Method for solution

3.1 Obtaining function words

Function words play an important role in this project, since all content words will be compared by how they behave relative to them. For the sake of simplicity, automatically acquiring these words was skipped and a list of the most common function words were selected for use throughout the rest of the project.

Rank	Function word	Rank	Function word	Rank	Function word	Rank	Function word	Rank	Function word
1	the	*11*	for	*21*	not	*36*	one	*61*	into
2	of	*12*	he	*23*	this	*37*	there	*65*	then
3	and	*13*	as	*24*	but	*39*	we	*74*	any
4	to	*14*	be	*25*	from	*43*	so	*89*	before
5	a	*15*	on	*27*	are	*44*	when	*112*	between
6	in	*16*	with	*28*	which	*45*	if	*118*	because
7	that	*17*	I	*30*	her	*51*	who	*124*	without
8	is	*18*	his	*33*	they	*57*	what	*129*	each
9	was	*19*	at	*34*	an	*58*	my	*132*	another
10	it	*20*	by	*35*	were	*59*	could	*145*	while

Table 3.1 – The selected 50 function words and their rank in the LOB corpus.

The table above contains the 50 function words chosen for this project. They are listed in order of rank from the LOB corpus. The top 20 words from the LOB corpus are captured in the list of function words. The majority of the top 30 are present, and admittedly, after that, they soon trail off to lower ranks. However, of the 50 picked, they all fall within the top 145 most frequent words in the LOB corpus, which still illustrates just how common the function words are, considering the corpus contains 50,000 distinct words.

Although obtaining function words without the aid of any expert linguistic knowledge would have certainly been preferable, the reason for this decision was due to the fact that size of the project was large enough – it was important to concentrate on the core objectives. If the results from using a given list of function words proved promising, then it would obviously be worthwhile as a future extension to implement an algorithm to perform this task.

3.2 Gathering distribution data

The method in which the distribution data is collected is the crux of this project and is what differentiates it from the many other pieces of research that have employed a clustering approach.

Firstly, a target function word and a target content word are selected. A window is specified, i.e., the distance (measured in number of words) either side of the target function word that you wish consider. For every occurrence of the target function word with in the corpus, the target content word is checked to see if it fell within the window. If it did, a note of the distance between the target words was made.

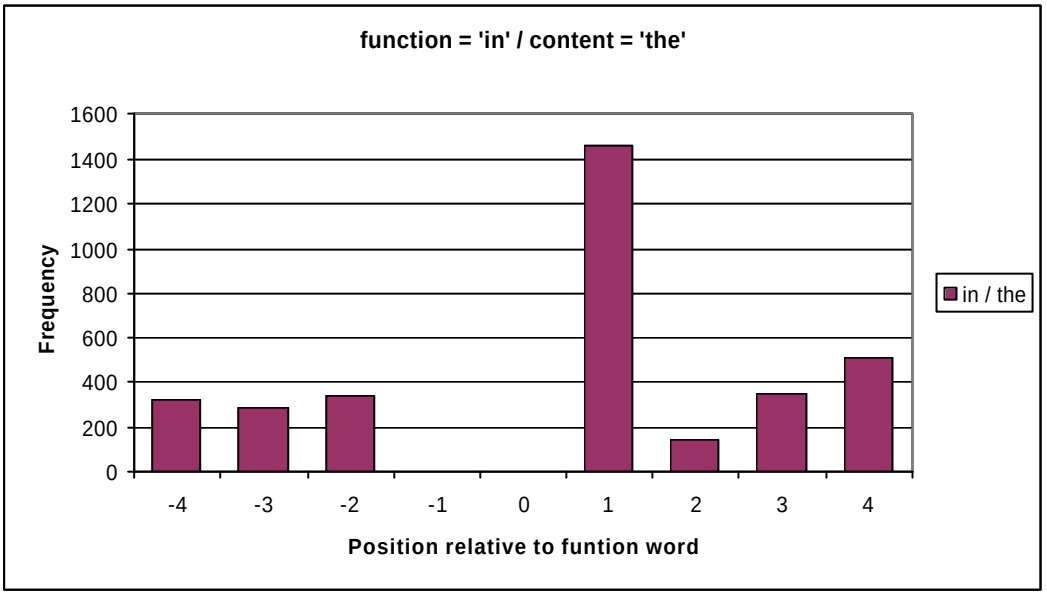


Fig. 3.1 – Graph showing how 'the' is positioned relative to 'in' from the Don Quixote corpus.

By treating each possible position of the target content word as a separate dimension, it is possible to represent the above graph as a vector by putting the frequency of occurrence for each dimension. To obtain the matrix for the above distribution:

Relative Position	*Frequency*
-4	317
-3	288
-2	341
-1	0
0	0
1	1457
2	146
3	350
4	510

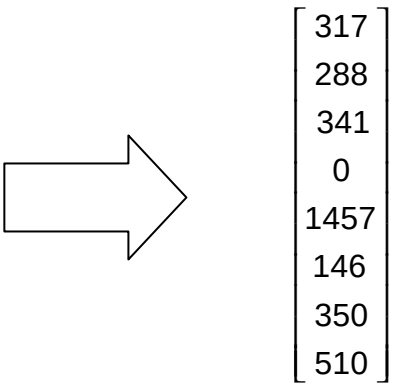


Fig. 3.2 – Demonstrating how the distribution data is translated in to a vector.

N.B. Storing the relative position of *0* in the matrix is not required since the content can never be at position *0*.

To produce a distribution profile for a given target content word against the function words, a vector can be created for each function word with the same content word, then all concatenated into a single vector (Fig 3.3). The number of dimensions that the final vector possesses will be number of function words multiplied by the window size.

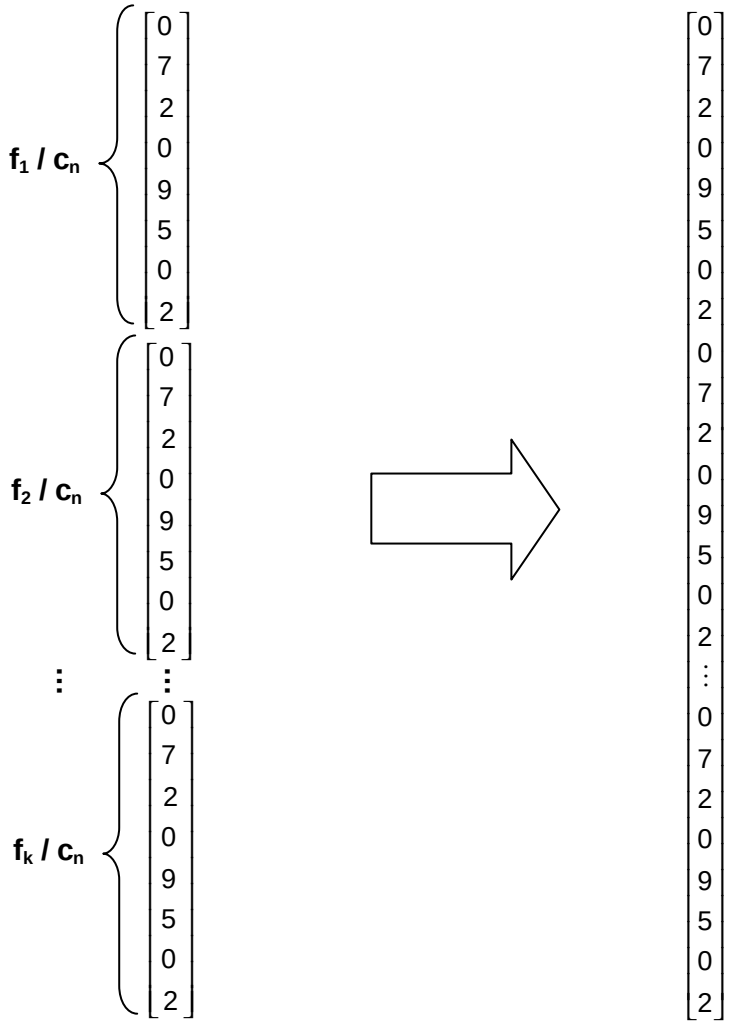


Fig. 3.3 – Joining the vectors to create a single vector that profiles the content word against all function words.

3.2.1 Normalising

In order to compare vectors fairly, they must first undergo normalisation. To illustrate why it is necessary, consider the following example: the distributions of the content words '*first*' and '*second*' relative to the function word '*the*'. They are both classed as the same part-of-speech – in this case, ordinals. Therefore, it would be reasonable to expect that they would behave similarly with the word '*the*'.

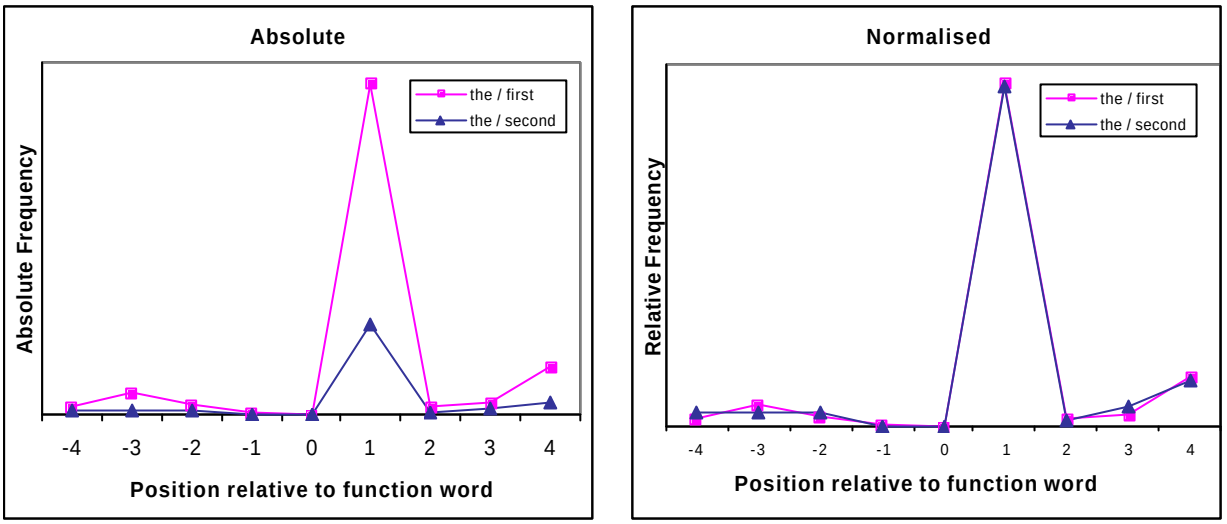


Fig. 3.4 – Two graphs plotted showing the effect of normalisation.

The distributions on the left of Fig. 3.4 show the absolute frequency distributions. It is clear to see that both content words share a similar 'shape'. However, because the content word '*first*' occurs more frequently in the corpus, the two words would be significantly dissimilar and therefore may not end up being classed together.

In the graph on the right of Fig 3.4. The data has been normalised and now the two distributions virtually overlap. By taking account of their relative frequency, there is a greatly increased chance that similar words will be shown to be so, regardless of their absolute frequency.

The formula for calculating a normalised vector is:

$$V_{ij}' = \frac{V_{ij}}{\sum_{j=1}^{m} Vij}$$

Where the j^{th} element of the vector V_i divided by the sum of all elements in V_i gives the j^{th} element of the normalised vector V_i'

3.3 Clustering

Using clustering techniques allows us to gather words into groups based on their distribution relative to the function words.

3.3.1 Measuring similarity

The advantage of using vectors to represent a content word's distribution is that they can be plotted as individual points in a high-dimensional vector space. And so, words that behave in a similar fashion will occupy a similar place in that vector space. It is therefore merely a matter of calculating the *distance* between two vectors which will give a measure of similarity[3].

The use of *distance* normally refers to the Euclidian distance between two vectors. However, there are many distance functions that can be used. For most problems, the generalised Minkowski distance is more than adequate:

$$D(V_x, V_y) = \left(\sum_{j=1}^{k} \left(V_{xj} - V_{yj} \right)^m \right)^{\frac{1}{m}}$$

Where k is the number of elements in vectors V_x and V_y. V_{xj} is the j^{th} component of the V_x vector. Changing the value of m gives different distance functions. Two such functions were used in this project.

3.3.1.1 Manhattan distance

Substituting 1 for m gives:

$$D(V_x, V_y) = \sum_{j=1}^{k} \left| V_{xj} - V_{yj} \right|$$

[3] For the sake of clarity, it is worth noting that distance is inversely proportional to similarity, i.e., the greater the similarity between two vectors, the smaller the distance between them.

This measure simply returns the magnitude of the difference between the two vectors.

3.3.1.2 Euclidian distance

Substituting 2 for *m* gives:

$$D(V_x, V_y) = \sqrt{\sum_{j=1}^{k} \left(V_{xj} - V_{yj}\right)^2}$$

3.3.2 Distance matrix

A distance matrix is used to store the distances between each pair of vectors. This matrix can be manipulated iteratively, which makes it an effective tool for the use in clustering.

$$\begin{array}{c} w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{array} \begin{bmatrix} 1.00 & & & & \\ 0.34 & 0.98 & & & \\ 0.83 & 0.12 & 0.88 & & \\ 0.01 & 0.39 & 0.72 & 0.45 & \\ 0.56 & 0.09 & 0.30 & 0.19 & 0.77 \end{bmatrix}$$
$$\begin{array}{ccccc} w_1 & w_2 & w_3 & w_4 & w_5 \end{array}$$

Fig. 3.5 – A toy example of a distance matrix.

The toy example shown in Fig. 3.5 demonstrates the role of a distance matrix. It is evident that the matrix need only be a lower triangle matrix. The reason for this is twofold. Firstly, the distance functions are commutative, i.e., $D(V_i, V_j) = D(V_j, V_i)$. Secondly, the distance between the same vector, $D(V_i, V_i)$, always equals zero. Only having to store half the matrix has clear computational benefits.

3.3.3 Clustering algorithms

This project makes use of agglomerative hierarchical clustering techniques. Generally, the procedure for clustering this method is as follows:

1. Calculate relevant statistics on the set of objects to be clustered in the form of vectors.
2. Where *N* is the number of objects, create a *N*x*N* matrix by measuring the distance between each pair of vectors.
3. Search for the pair which are closest (i.e., has the lowest value) as this is the similar word pair.
4. Calculate a new distance matrix, replacing the most similar pair with a measurement which represents the union of them. Thus, the new matrix will be one row and one column smaller in size.
5. Continue to search for the lowest value and then recalculate until the matrix has merged into a single cell.

The way in which the distances between clusters are defined, during step 3, is what distinguishes one clustering algorithm from another (Everitt 1993).

Clustering would be computationally expensive if it were not possible to calculate new groupings iteratively. Fortunately, it is possible to determine new clusters from the distances in the previous distance matrix. Lance and Williams (1967) demonstrated that many clustering algorithms can be derived from a single generalised equation:

$$D_{P,Q} = a_X D_{X,Q} + a_Y D_{Y,Q} + b D_{X,Y} + g \left| D_{X,Q} - D_{Y,Q} \right|$$

where the parameters a, b, and g correspond to a clustering method. $D_{P,Q}$ is the distance between clusters *P* and *Q*. The cluster *Q* is the new cluster formed by the merging of clusters *X* and *Y*.

Of the many methods that exist, four were implemented. These are described below.

3.3.3.1 Complete Linkage

Also know as the '*furthest neighbour*' method since it measures the distance between two groups as the most distant pair of individual objects, one from each group.

The parameters for complete linkage are: $a_X = 0.5$, $a_Y = 0.5$, $b = 0$ and $g = 0.5$. Which gives:

$$D_{P,Q} = \frac{D_{X,Q}}{2} + \frac{D_{Y,Q}}{2} + \frac{\left|D_{X,Q} - D_{Y,Q}\right|}{2}$$

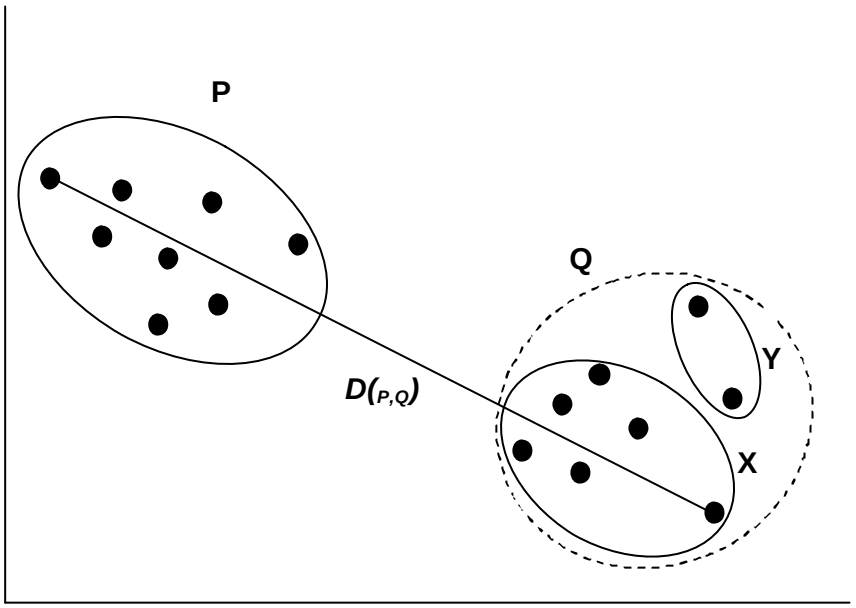


Fig. 3.6 – Diagram illustrating Complete linkage, or '*furthest neighbour*'. The new cluster *Q* is formed from combining the two groups *X* and *Y*.

3.3.3.2 Group-Average

The group-average method measures distance by taking the average of the distances between all pairs of individual objects from the two groups.

The parameters for group-average are: $a_X = \frac{N_X}{N_P}$, $a_Y = \frac{N_Y}{N_P}$, $b = 0$ and $g = 0$. Which gives:

$$D_{P,Q} = \frac{N_X D_{X,Q}}{N_P} + \frac{N_Y D_{Y,Q}}{N_P}$$

N_X and N_Y are the number of objects in the clusters X and Y respectively. Also,
$N_P = N_X + N_Y$

3.3.3.3 Weighted Group-Average

A slightly modified version of the group-average method. This method ignores the size of the clusters to be grouped in favour of the assumption that they are of equal size. The reason for this approach is to give smaller clusters a greater influence when grouped with a much larger cluster.

The parameters for weighted group-average are: $a_X = 0.5$, $a_Y = 0.5$, $b = 0$ and $g = 0$. Which gives:

$$D_{P,Q} = \frac{D_{X,Q}}{2} + \frac{D_{Y,Q}}{2}$$

3.3.3.4 Ward's Method

Ward's method (Ward 1963) differs somewhat from the algorithms described above. The distance calculated has no real relevance in terms of *geometric* distance, rather it is a statistical measurement of the minimal *information loss*. The method works as follows: the central point for pairs of clusters is evaluated. The total sum of squared distances from this central point to all objects in this hypothetical cluster is then calculated. The cluster with the smallest sum of squares is the new cluster.

Zupan (1982) regarded Ward's method as "a very efficient clustering method, but favours the grouping of small clusters." What he may have thought a potential disadvantage could be beneficial for the clustering of words.

The parameters for Ward's method are: $a_X = \frac{N_{XQ}}{N_{PQ}}$, $a_Y = \frac{N_{YQ}}{N_{PQ}}$, $b = -\frac{N_Q}{N_{PQ}}$ and $g = 0$.

Which gives:

$$D_{P,Q} = \frac{N_{XQ}D_{X,Q}}{N_{PQ}} + \frac{N_{YQ}D_{Y,Q}}{N_{PQ}} - \frac{N_Q D_{X,Y}}{N_{PQ}}$$

N_a is the total number of objects in cluster *a*.
N_{ab} is the combined total of objects in clusters *a* and *b*.

3.3.3.5 Other algorithms

Four other algorithms were in fact implemented for this project. They were: Single Linkage, Median, Centoid and Centre of Gravity. Unfortunately, they performed so poorly for this particular application that they were not considered in the evaluation.

3.4 Obtaining a corpus

Choosing a corpus should never be a trivial task as they can influence greatly the performance of a system. For example, if a speech recognition system were being developed, then it would benefit from a corpus of spoken language to train the system, rather than any other types. It does not mean that a corpus of written language is not valid, however, it merely does not reflect the most appropriate usage.

Another important feature of a corpus is its size. The major reason for this is that it combats the data sparseness problem. A common approach for researchers in the past few years has been to compile massive corpora by extracting text from USENET newsgroups. Examples include:

- Hughes (1994) collected 35 million words, of which 30 million were taken from USENET.
- Finch and Chater (1992) built a 40 million word USENET corpus.
- Lund and Burgess (1996) experimented with a 160 million word USENET corpus.
- Levy and Bullinaria (2001) managed to accumulate a vast 168 million word corpus from USENET.

Despite the availability of such vast bodies of written text, it may be unwise to rely on them, and their ever increasing size. If a machine learning system was developed in order to penetrate the syntax of an unknown language, it may fail if its success depends on analysing massive samples of that language. In fact, to collect 168 million words in any language other than English would be not be a trivial task. Corpora of other *known* languages are not nearly as plentiful, or as large. USENET cannot be relied upon due to the dominant language being American-English.

For experimentation with the English language, two corpora were used, both chosen to be deliberately small (compared to modern day corpora). The first corpus was taken from a

single source; an English translation of *Don Quixote de la Mancha* by Miguel de Cervantes (Ormsby 1885). An electronic version was obtained freely from Project Gutenberg[4]. This body of text contains 426,700 words, consisting of 16,000 unique tokens. The size of this corpus, though relatively small, is still adequate to perform the distributional analysis on, and cluster effectively. The added benefit is that execution time for the whole clustering process is greatly reduced due to not handling a massive number of words.

The second corpus used was the LOB corpus. It totals one million words, of which there are approximately 50,000 unique tokens. At over twice the size of the *Don Quixote* corpus, it should provide an interesting glimpse at how the size of corpus can affect the end performance of the clustering process.

The clustering process will not attempt to cluster all of the words in the entire corpus. For the experiments performed in this work, a limit of the most frequent 500 content words was set. Such a limit may sound small relative to the number of distinct words in the corpora, however, if you recall Zipf's Law from section 2.6, the vast majority of the words in a body of text can be accounted for from a small percentage of the highest ranked words.

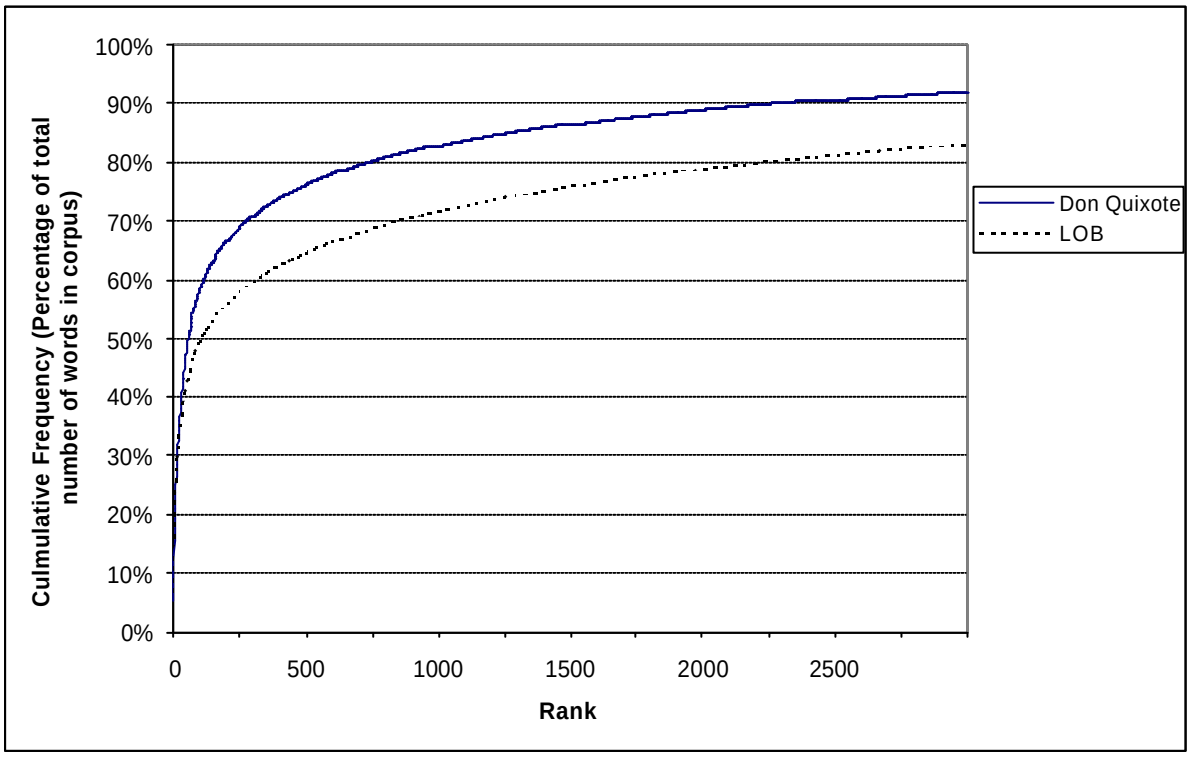


Fig. 3.7 – Cumulative frequency graphs plotted for both corpora.

[4] An online resource of royalty-free e-texts. http://promo.net/pg/

It can be seen from Fig. 3.7 that the 500 most frequent words for both corpora captures a high percentage of the total number of words. Indeed, a coverage of 77.1% is attained for the *Don Quixote* corpus, and a slightly lower 64.4% for the LOB corpus. Therefore, for the purposes of any experiments undertaken, only clustering 500 words is acceptable.

For experimentation with an alternative language, Spanish was selected. Acquiring a relatively small Spanish corpus proved to be straightforward, as an electronic copy of the original Spanish version of *Don Quixote* was again obtained courtesy of Project Gutenberg. An interesting observation with the Spanish *Don Quixote* is that it is only 383,200 words in size, with a vocabulary of 24,000 words. This is roughly a difference of 43,000 words compared to the English translation. Yet, the vocabulary is 8,000 richer than the English equivalent. The variation in the word count can largely be attributed to the translator's preface which totals to over 17,000 words. The rest of the difference illustrates that there is never a direct one-to-one relationship between words from different languages. In this example, there are clearly words and phrases in Spanish that cannot be expressed in the same number of words for their English equivalent.

4 Design and Implementation

This section provides an overview of the software implemented and discusses the design of the algorithms developed for this project. The set problem was substantial, therefore implementation of the solution was broken up into smaller, more manageable sub-tasks. This is not unusual programming practice. The end result is a suite of tools - some of which are useful on their own – that when used together, perform the clustering as required by the project.

Fig. 4.1 shows the entire process from the original corpus to the end result. The corpus is processed by a number of different tools, and the clustering stage comes at the very end of a relatively long route.

4.1 Segmentation

The main role of this tool is to isolate each word from the raw text of the inputted corpus. To do this, firstly, (almost) all punctuation is removed, since there is no information to be gained from keeping it. Secondly, all upper case characters are converted to lower case to rule out potential problems with case sensitivity.

4.1.1 Punctuation policy

A more detailed description of the way punctuation was dealt with is as follows:

1. Read a ‘word’ (a string delimited by a space) from the input corpus.
2. If the word begins with any punctuation marks, e.g., “ ‘ ([{ , then strip from the word.
3. If the word contains a hyphen or a slash, then the characters either side are separated into two individual words.
4. If the penultimate character is an apostrophe, then *do not* remove it. This preserves two-word contractions such as *I’m* and *don’t*.
5. If word ends with punctuation marks, e.g., ? ! , . : ; “ ‘)] } , then strip from the word.
6. Finally, with the resultant word, convert any uppercase letters to lowercase.

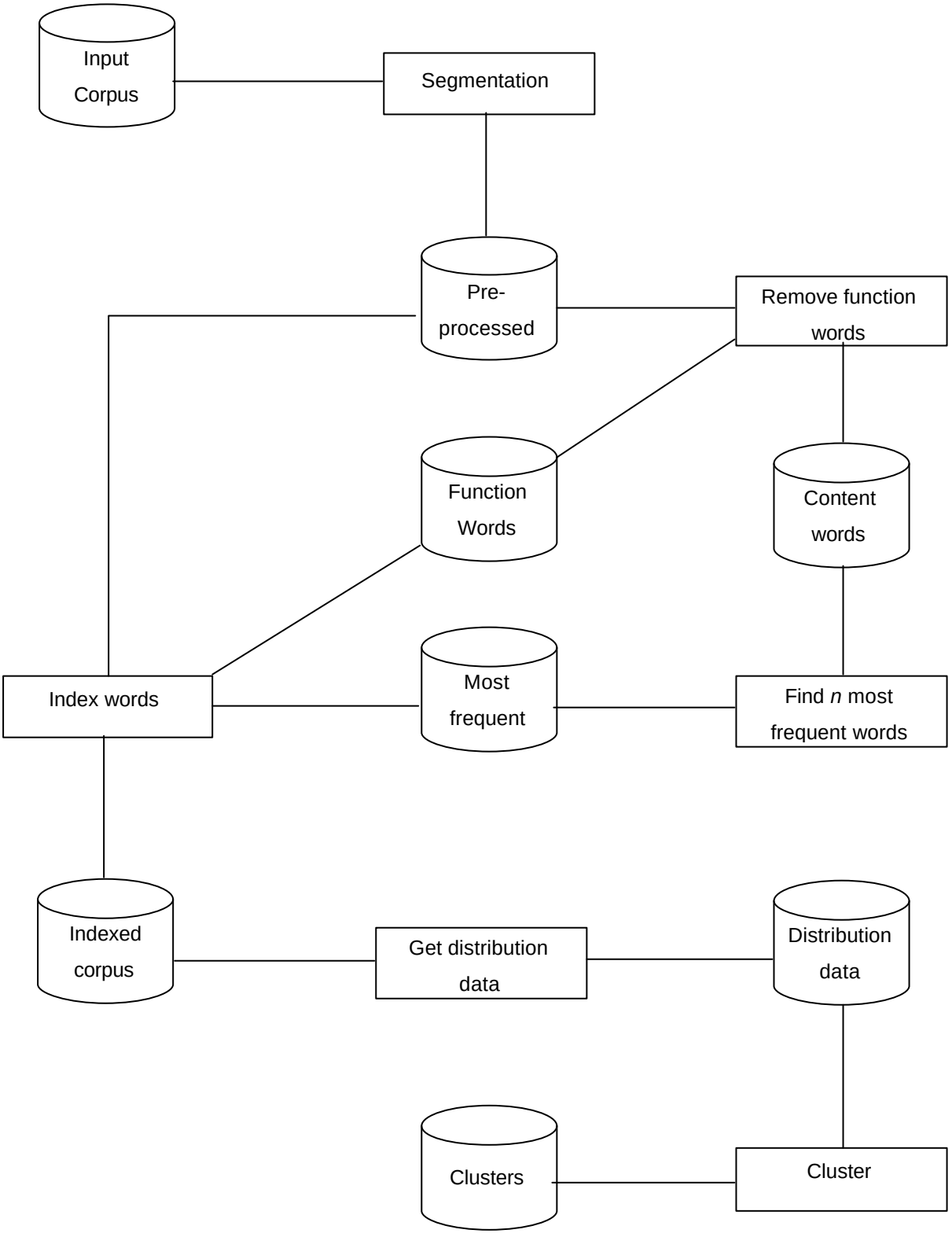


Fig. 4.1 – Showing the flow of data through the system and how each tool interacts.

4.2 Remove function words

This program takes a corpus (pre-processed by the program described above) and a specified list of function words, then simply works by the following steps:

1. Read in next word.
2. If word is a specified function word then do nothing.
3. Else output word .

Repeat the procedure until all words from corpus have been read. The output from this program is a corpus of content words, since all the specified function words have been filtered out. The purpose for this tool may not be entirely obvious, but the output is used by other tools later in the pipeline that require just the content words of the original corpus.

4.3 Find *n* most frequent words

This tool, quite plainly, will find the most frequent words in a given body of text. The number of words required is specified by the user, and the output is a list of words with their absolute frequency of occurrence within the text, sorted in descending order of frequency (i.e., the highest frequency first). The program was originally to be written from scratch. Fortunately, Hughes (1994) described a method of combining a number of core UNIX tools to perform this task. As a result, the tool written for this project was largely inspired from Hughes' method to avoid unnecessary reengineering.

By inputting the corpus with function words filtered out, this tool will give the most frequent content words in the corpus. This is necessary because the entire corpus will not be clustered, only those that occur often enough as they contain some valuable information.

4.4 Indexing

The use of indexing is purely for efficiency – at least in the context of this project. This tool, given a list of words to index, will search through a given corpus and record the position of each occurrence of the specified words.

To illustrate with a toy example on the following sentence: “the cat sat on the mat”.

Word	the	cat	sat	on	the	mat
Position	1	2	3	4	5	6

Word	cat	mat	on	sat	the
Index	2	6	4	3	1, 5

Fig. 4.2 – Demonstrating how words are indexed.

The above example may not look very impressive, but word indexes have a variety of applications, including information retrieval in the form of *inverted files*, and *concordance*[5]. For the purposes of this project, the word index makes the process of gathering the distributional data of a given word much more efficient.

4.5 Distribution

This is the program that gathers the all-important distribution information – the context measurement for the subsequent clustering process. For the program to function, it requires an indexed list of function words, and an indexed list of content words. It then processes each possible combination of function word and content word in the following manner:

```
For each content word, c
  For each function word, f
    For each occurrence of c, at position x
      For each occurrence of f, at position y
        If x is inside the window of y then record its relative position
```

The *window* refers to the number of words either side of the target word that is to be considered when recording the occurrences of the target content word. In the examples used in section 3, the window was ±4.

[5] “Comprehensive listing of a given item in a corpus (most often a word or a phrase), also showing its immediate context.” (Oakes 1998)

Using the word index created in Fig. 4.2, with a window of ±4, let the target function word be *the* and target content word be *cat*.

Relative position	-4	-3	-2	-1	1	2	3	4
Freq	0	1	0	0	1	0	0	0

Fig. 4.3 – Collecting distribution data from word indexes.

4.6 Clustering

The clustering program involves a number of important steps. After reading the distribution data produced from the previous step, it must first convert that data into vectors as demonstrated in section 3.2. The data from the distribution program is ordered by content word, and then by function word, which makes creating the function word profile vector for each content word much simpler.

The distance matrix is created, and each vector is compared using a specified metric. Once the distance matrix is filled, the clustering using a specified algorithm can commence. Clustering will continue to group words together until the predetermined number of clusters is reached.

On completion of the clustering, the clusters are outputted for the user.

4.7 Choice of programming language

The choice of programming language should not alter the actual functionality of the software. However, it may determine the methodology employed to solve the problem in terms of design and implementation. The main categories are procedural languages, functional languages, logical languages and object-orientated languages. *C++* was used to implement the algorithms in this project, which falls somewhere in between a procedural and object orientated language[6]. The pros and cons of each programming language shall not be discussed here. Needless to say, reasons for its selection include: its ability to cope easily

[6] *C++* is essentially the procedural language of *C*, with object-orientated capabilities added on top. To some, such a recipe leaves *C++* unfavourable, to others it is beneficial as it gives the best of both worlds.

with numerical manipulation; can handle large data structures in memory; execution speed of *C++* code compared other popular languages for NLP such as *Perl* and *POP-11* (due to their suitability for symbolic manipulation) is considerably quicker since it is compiled into machine code, rather than run by an interpreter.

5 Experiments

5.1 Factors

The following factors can be experimented with:

- Type of corpus
- Size of the corpus
- Number of function words
- Number of content words to be clustered
- Size of the window
- Metric used
- Clustering algorithm used
- Number of resulting clusters

There is clearly plenty of scope for research into how the above factors affect the overall performance of the clustering. However, due to time restrictions, only a selection of the above factors will be investigated.

As discussed in section 3.4, the English corpora for this project are the 427,000 word *Don Quixote* text and the one million word LOB corpus. The first two factors will remain static with these two corpora and their respective sizes. The number of content words will stay fixed on 500.

5.2 How to evaluate clusters

It is first necessary to describe the method of evaluation in order to understand how the performance of the clustering was measured. Without a reliable approach, it would not be possible to quantify the effects of a given factor.

Hughes (1994) outlined an excellent approach to automatically evaluating the clusters his algorithms produced. He used a pre-tagged LOB corpus, where all words had been assigned an appropriate part-of-speech. However, the LOB tagset comprises of over 130 separate tags, which was too detailed, therefore, he devised the *reduced LOB tagset*. This shrinks the tagset to only 23 parts-of-speech.

Reduced Tag	Replaced Tags	Type of Item
ADJ	J*	Adjective
ADV	R*	Adverb
ART	A*	Article
CCON	CC*	Coordinating conjunction
CARD	CD*	Cardinal numeral
DET	DT* PP$*	Determiner
EX	EX	Existential *there*
EXPL	U*	Interjection
LET	Z*	Letter of the alphabet
MD	MD	Modal auxiliary verb
NEG	XNOT	Negator
NOUN	N*	Noun
ORD	O*	Ordinal numeral
OTH	&*	Foreign words, formulas
PAST	BED DEBZ BEN DOD HVD HVN VBD VBN	Past tense verb
PREP	I*	Preposition
PRES	BE BEG BEM BER BEZ DO DOZ HV HVG HVZ VB VBG VBZ	Present tense verb
PRON	P* (not PP$*)	Pronoun
PUNC	! ( ) , . … ; : ? *..	Punctuation
QUAL	Q*	Qualifier
SCON	CS*	Subordinating conjunction
TO	TO	Infinitive marker
WH	W*	WH-word

Table 5.1 – The reduced LOB tagset. Note * is a wildcard character, e.g., J* means any tag beginning with the letter J and *any* letter after.

The automatic approach to evaluate a given cluster works as follows:

1. For each word, lookup in the LOB corpus and find any tags that it has assigned to it (from the reduced tagset).
2. Determine which is the most common tag for that cluster.
3. Calculate the ratio of the number of words in the cluster that possess the most common tag, to the total number of words in the cluster. Return as a percentage.

The *Don Quixote* corpus contains words that do not appear in the tagged LOB corpus. For any word that does not appear in the LOB corpus, then it is simply assigned "*UNK*" to represent that its part-of-speech is unknown.

5.3 Effect of clustering algorithm

It was anticipated that the clustering algorithm would have a profound effect on the eventual clustering accuracy. Each algorithm behaves differently, therefore it was a reasonable assumption to make, and was backed up by this experiment: number of clusters = 100; number of function words = 25; window size = 12.

Corpus	Metric	Algorithm			
		Complete linkage	Group average	Weighted group average	Ward's method
LOB	Euclidian	82.2%	86.5%	83.6%	77.2%
	Manhattan	80.9%	83.5%	85.6%	77.1%
Don Quixote	Euclidian	74.9%	78.2%	77.2%	77.2%
	Manhattan	73.9%	84.5%	82.7%	68.7%

Table 5.2 – The effect of the clustering algorithm on the clustering accuracy.

Perhaps more interesting than the varying nature than the algorithms themselves, is the role that the metric plays. In Fig. 5.1, the effect of the metric is illustrated clearly, and can alter the performance of the subsequent clustering considerably. There is a difference of 8.5% between the two metrics when used with Ward's method. It is not entirely obvious why this is the case. For the larger LOB corpus, Ward's method performs almost equally with either metric. In fact, differences between the metrics are much less pronounced in the LOB corpus, which suggests that the size of the corpus is influencing the overall performance.

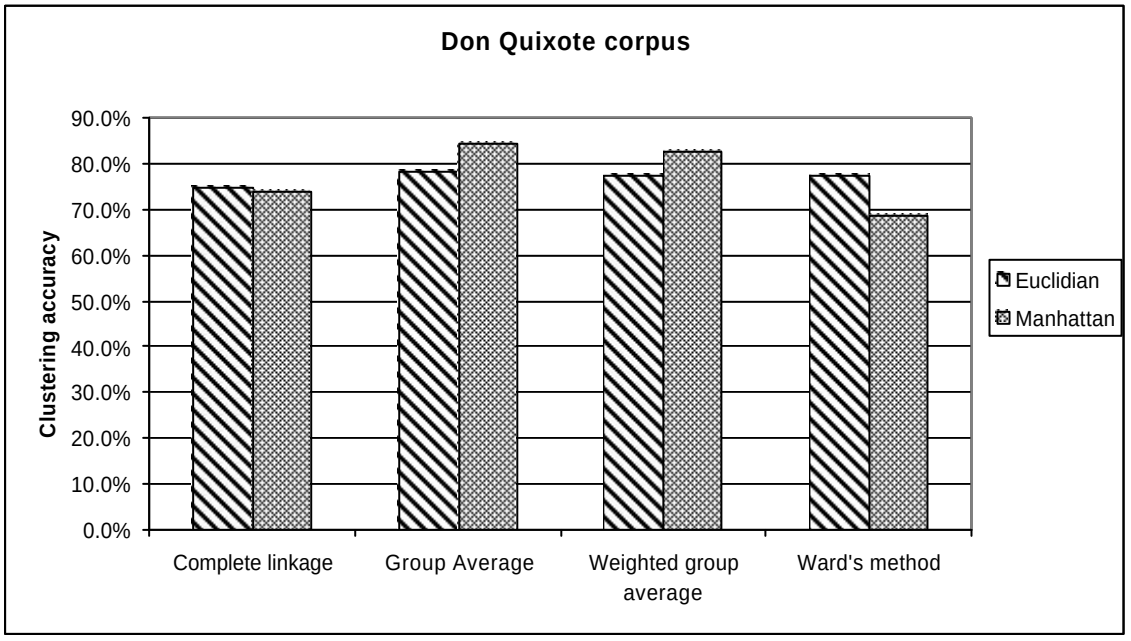


Fig. 5.1 – A graph plotting the effect of the clustering algorithms for the *Don Quixote* corpus.

Regardless of the metric, overall, Group Average performs the best of the four algorithms used. Closely followed by Weighted Group Average, Complete linkage and finally Ward's method. This order nearly always remains for any given cluster size, number of function words and window size. There are a few exceptions to the rule, where for example weighted group average may perform slightly better than group average.

5.4 Effect of number of function words

Due to much of the project revolving around the function words, it makes sense to investigate the role they play. Fifty function words were selected (see Table 3.1); the *n* highest ranking function words were used, and the distribution of the target content words were profiled against those *n* words.

The first experiment looked at the relationship between the clustering algorithm employed, and the number of function words. Fig. 5.2 shows the results for an experiment where: corpus = *Don Quixote*; number of clusters = 100; window size =12; metric = Euclidian.

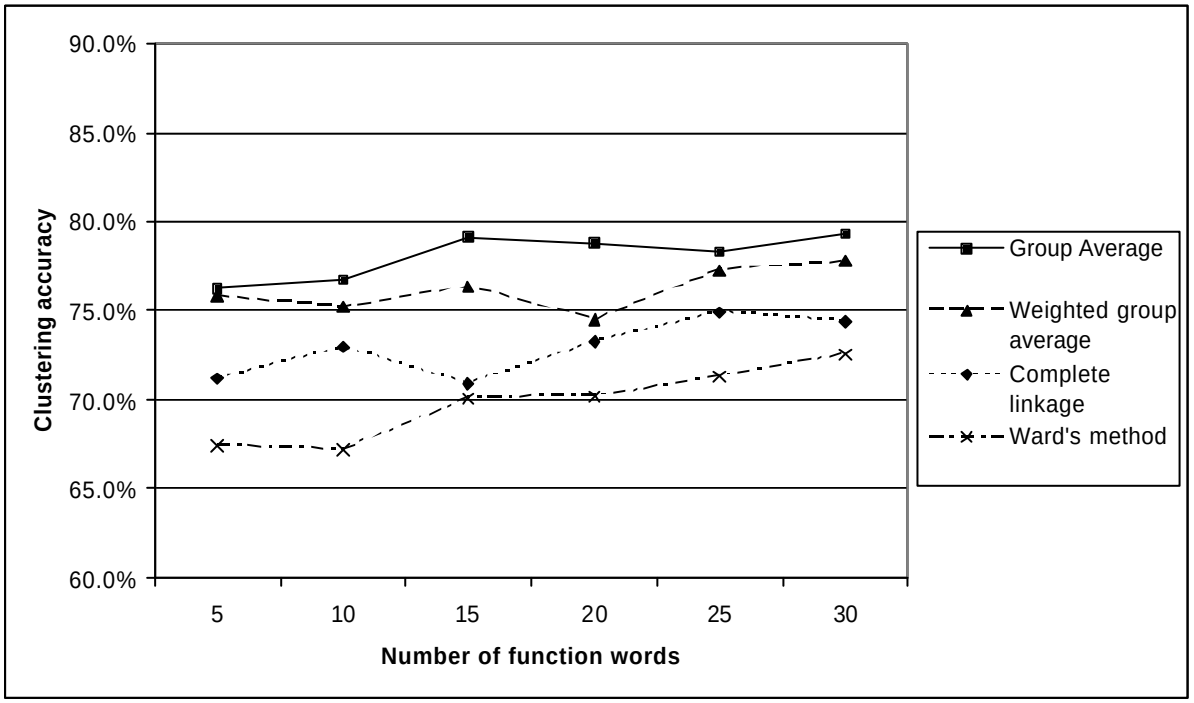


Fig. 5.2 – Graph plotting the relationship between the clustering algorithm and the number of function words used for the *Don Quixote* corpus.

There is a trend that suggests the greater the number of function words used, the better the clustering accuracy. However, it is clearly not a smooth slope. In fact, the relationship is much more erratic if the corpus is switched to LOB (see Fig. 5.3).

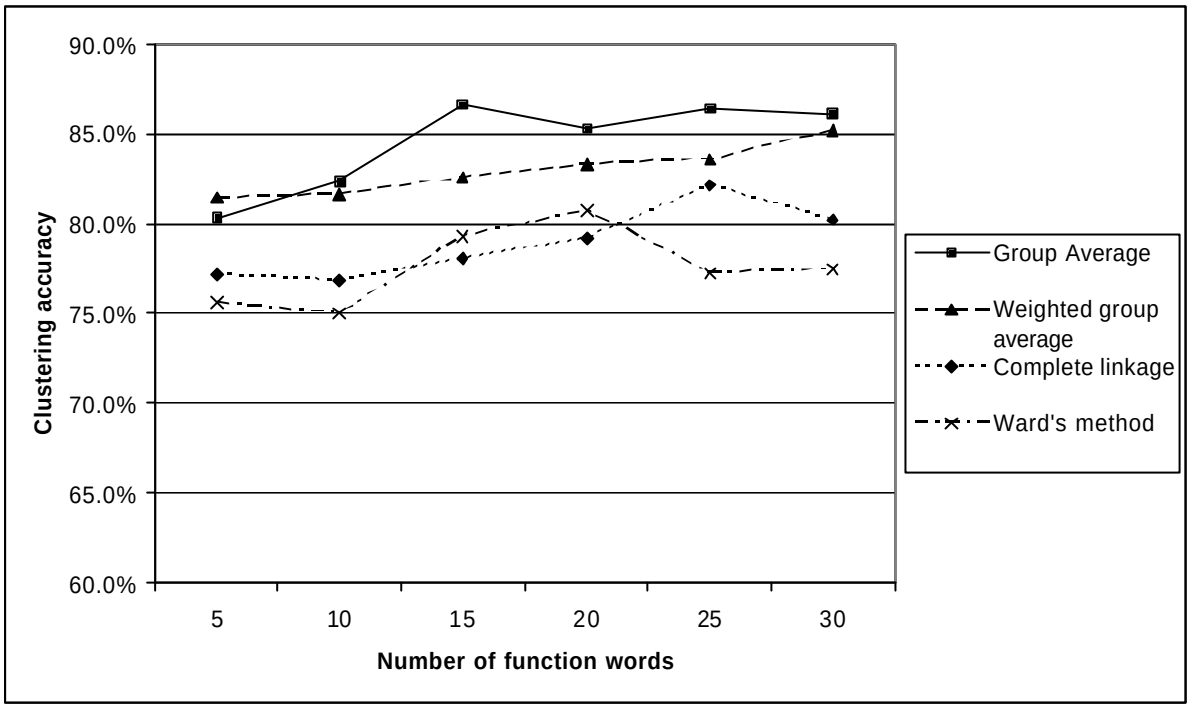


Fig. 5.3 - Graph plotting the relationship between the clustering algorithm and the number of function words used for the LOB corpus.

Focusing on the relationship between the number of function words and the window size are related. Table 5.3 shows the results from an experiment where: number of clusters = 100; clustering algorithm = Group Average; metric = Manhattan.

Num. of function words	DQ			LOB		
	Window size			*Window size*		
	4	*8*	*12*	*4*	*8*	*12*
5	72.0%	73.9%	78.3%	77.0%	79.9%	80.6%
10	72.5%	76.5%	81.7%	80.7%	83.3%	84.7%
15	74.8%	75.5%	81.1%	81.7%	84.4%	86.3%
20	75.8%	77.7%	82.7%	84.1%	85.5%	87.0%
25	75.4%	78.2%	84.5%	83.0%	85.6%	83.5%
30	77.3%	78.1%	83.6%	85.1%	85.5%	86.7%

Table 5.3 – The clustering accuracy for number of function words against the window size.

The results show again that in general, for any window size, the greater the number of function words used, the better the accuracy. Fig. 5.4 shows the above results for the LOB corpus in a visual form, to give a better feel of the behaviour. Initially, the performance increased substantially for each increment of 5 function words, however, it slows down by 20 function words. Performance in fact dips when the number of function words is at 25. The decline in performance looks worse than it really is from the graph due to the scale of the *y*-axis. It is a mere one percent.

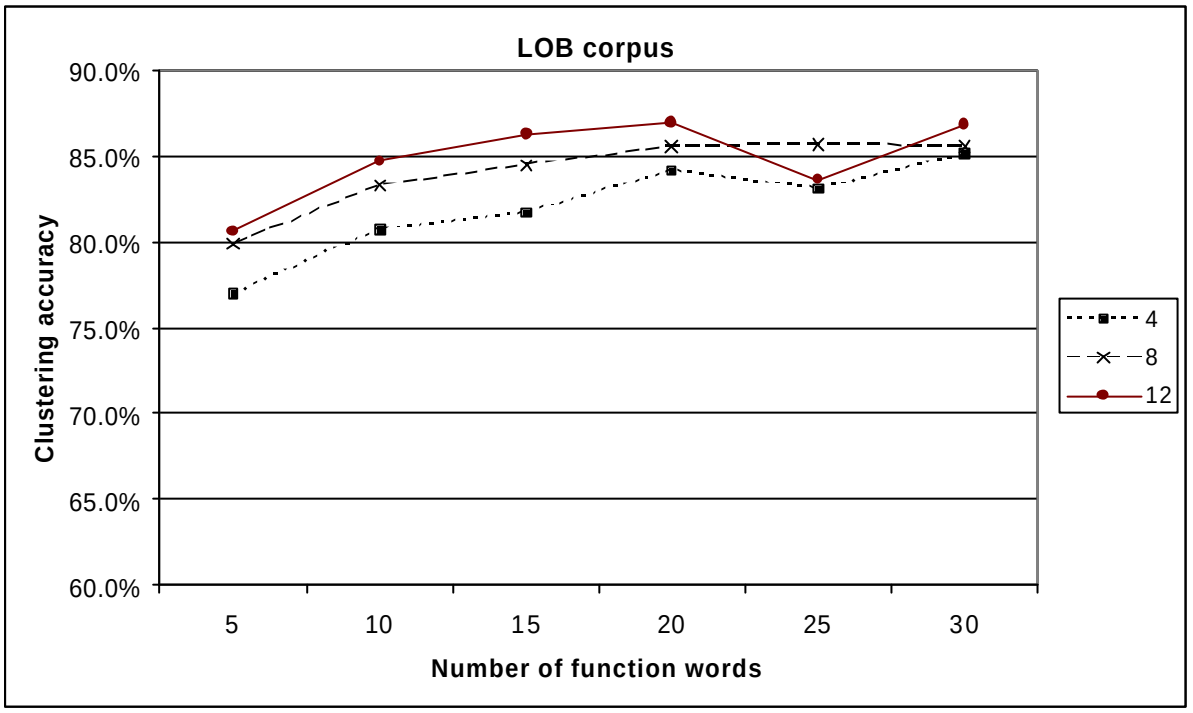
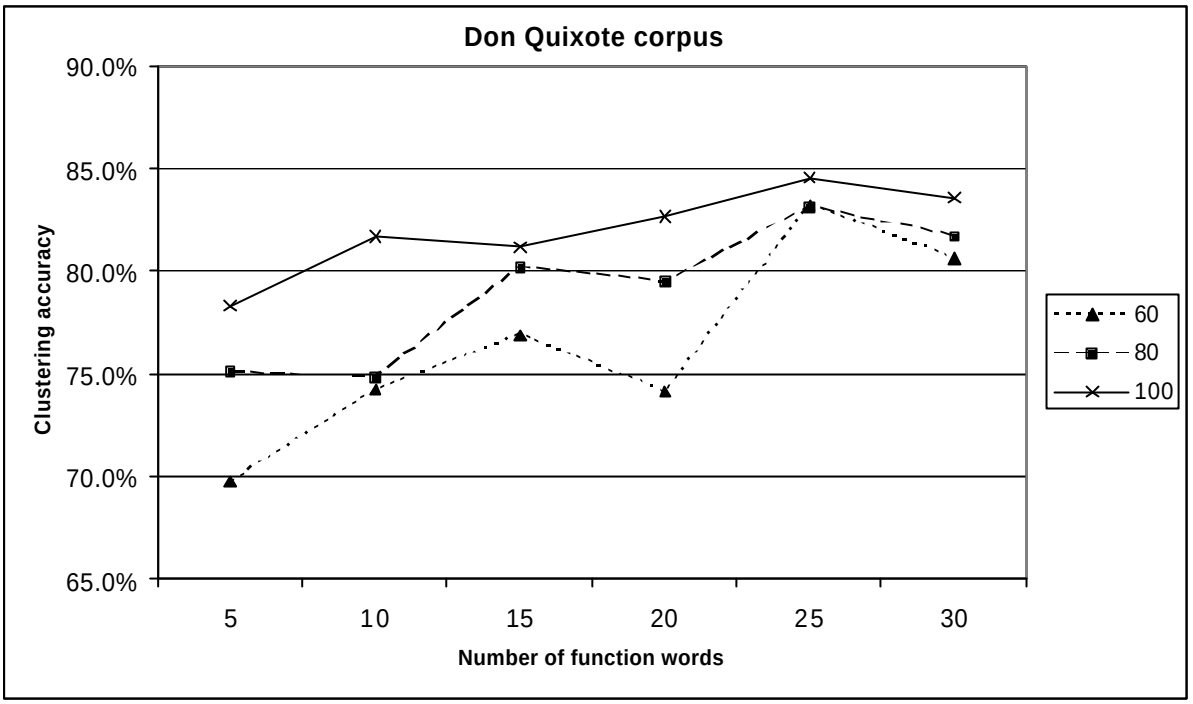


Fig. 5.4 – Graph showing how the number of function words affects the clustering accuracy with respect to the window size used, for the LOB corpus.

Finally, investigating the effect of the number of function words for different numbers of clusters. Unfortunately, only a weak link was found. As Fig. 5.5 shows (number of clusters = 100; window size = 12; clustering algorithm = Group Average; metric = Manhattan; corpus = *Don Quixote*), performance was rather irregular (even more so with the LOB corpus), thus a reliable conclusion as to how these two factors behave together cannot yet be obtained. It does *seem* to increase as the number of function words increases. The difference between the performance for five and thirty function words is considerable. However, with so many peaks and troughs in between, it suggests that the number of resulting clusters has a greater influence on the overall performance which is why this experiment returns vague results.



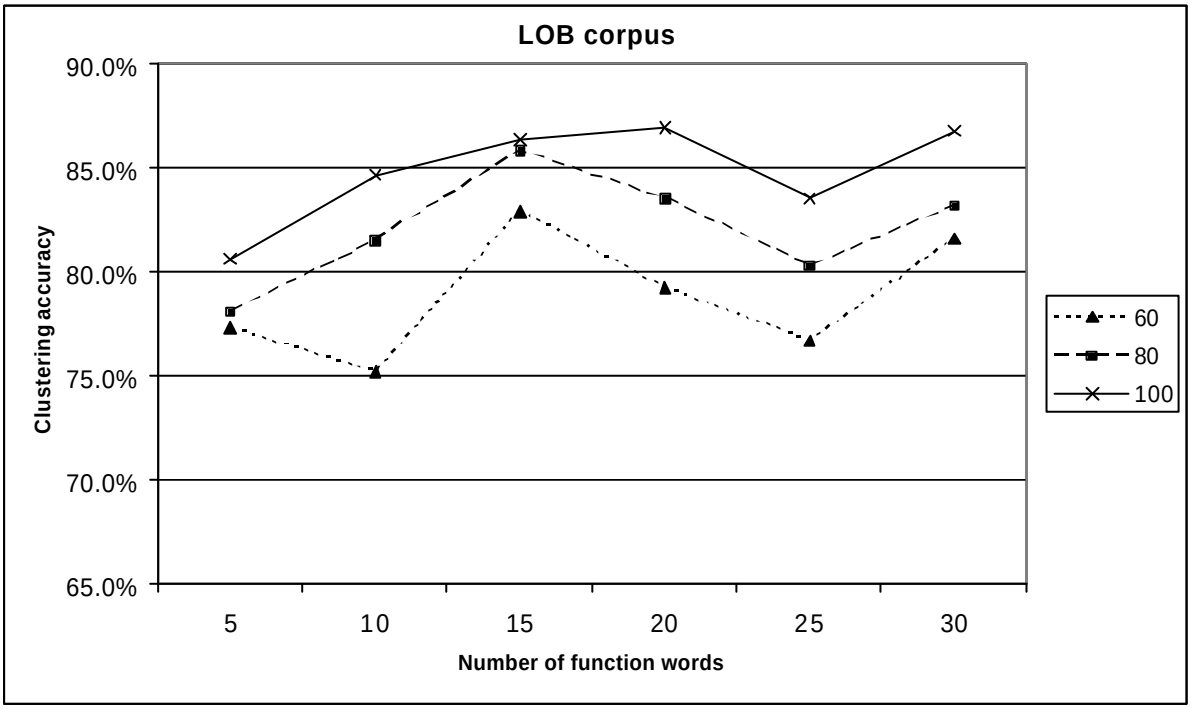


Fig. 5.5 – Graphs plotting the relationship between the number of function words and number of resulting clusters.

5.5 Effect of window size

The effect of varying the window size can be demonstrated from the following experiment, as shown in Table 5.4, where: Number of clusters = 100; number of function words = 25. The table is quite large as it shows the different window sizes for each clustering algorithm, with each metric for both corpora. Looking along each row, it is easy to see that generally, as the window size increases, the clustering performance improves.

Corpus	Metric	Algorithm	Window size					
			2	*4*	*6*	*8*	*10*	*12*
D.Q.	Euclidian	*Complete linkage*	71.3%	72.2%	71.8%	73.6%	72.2%	74.9%
		Group Average	74.1%	76.0%	77.5%	78.4%	79.2%	78.2%
		Weighted group average	73.5%	75.3%	74.7%	75.6%	76.5%	77.2%
		Ward's method	69.8%	69.8%	70.2%	71.6%	70.6%	71.3%
	Manhattan	*Complete linkage*	72.3%	73.9%	74.7%	71.4%	69.9%	73.9%
		Group Average	74.7%	75.4%	75.2%	78.2%	79.1%	84.5%
		Weighted group average	75.2%	75.0%	72.7%	76.2%	78.8%	82.7%
		Ward's method	70.4%	71.0%	72.4%	70.4%	70.5%	68.7%
LOB	Euclidian	*Complete linkage*	75.4%	77.6%	78.4%	79.3%	80.1%	82.2%
		Group Average	81.0%	81.4%	84.2%	85.3%	87.3%	86.5%
		Weighted group average	80.3%	81.1%	83.6%	84.1%	82.8%	83.6%
		Ward's method	74.8%	72.7%	77.5%	78.4%	78.3%	77.2%
	Manhattan	*Complete linkage*	80.3%	79.3%	80.1%	80.8%	82.5%	80.9%
		Group Average	83.6%	83.0%	83.9%	85.6%	86.7%	83.5%
		Weighted group average	82.7%	83.1%	84.9%	85.2%	82.6%	85.6%
		Ward's method	76.1%	76.6%	78.0%	79.7%	79.6%	77.1%

Table 5.4 – The clustering accuracy for different window sizes.

As with all the factors investigated so far, it is not a steady, linear increase. The change in accuracy can be uneven. This is depicted more visibly in Fig. 5.6.

5.6 Effect of the number of clusters

This factor was the easiest to predict. At risk of sounding like a scratched record, it was expected that the greater the number of clusters, the better the clustering accuracy. If the number of clusters was equal to the number of content words to be clustered, then the performance of the clustering would be 100% accurate.

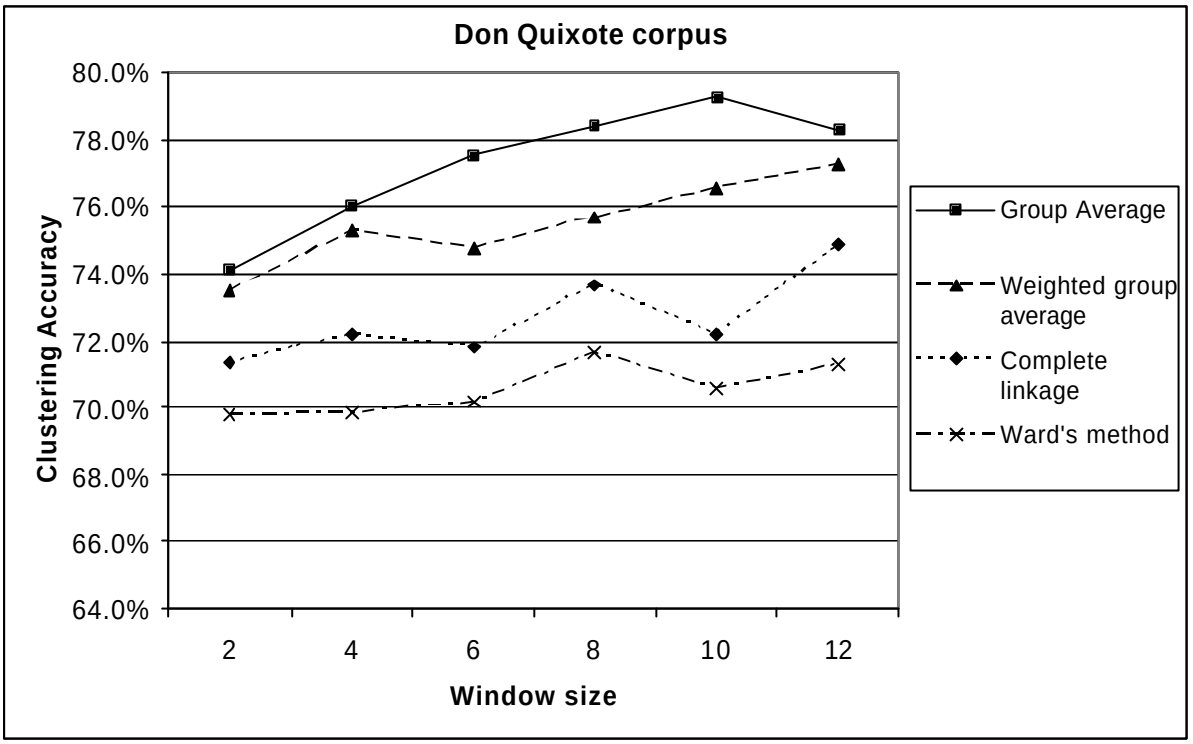
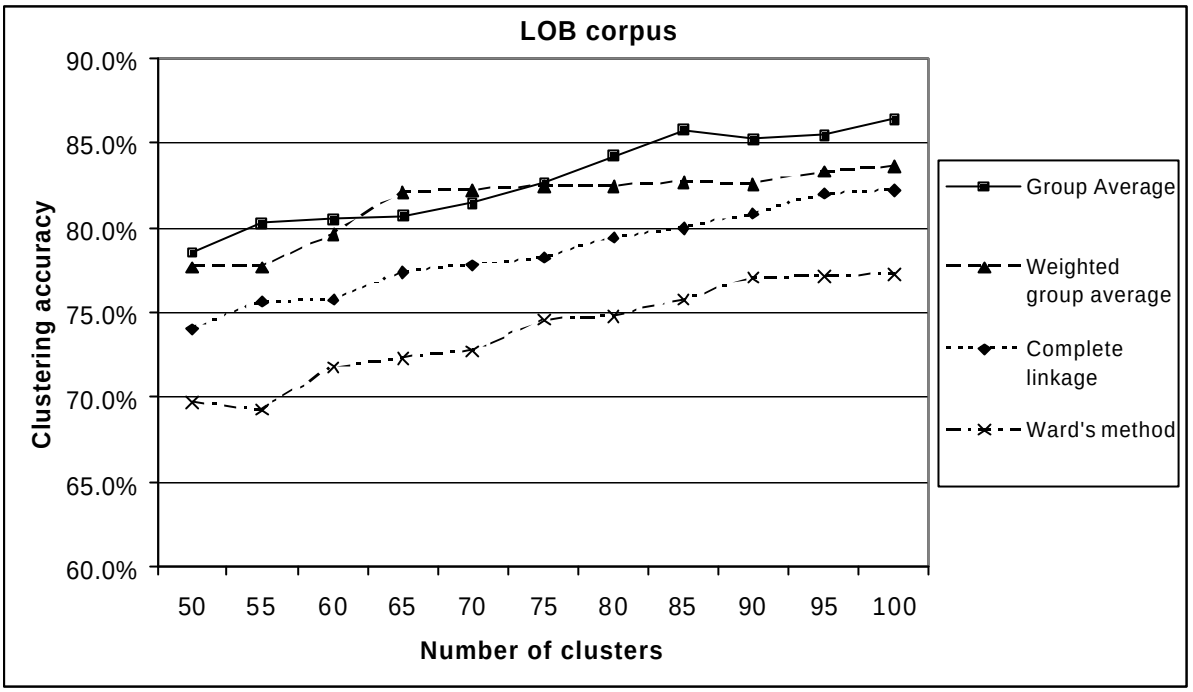


Fig. 5.6 – The effect of the window size for the different clustering algorithms. (Metric = Manhattan)

That is because each cluster is simply an individual word and so the method to automatically evaluate a cluster will always return 100%. Conversely, if the number of resulting clusters was simply a single cluster, then the accuracy would be very poor. The one cluster contains all the content words to be clustered, and cannot all be satisfied with a single part-of-speech that will be assigned to it by the automatic evaluation.

The prediction was confirmed with the following experiment, where: number of function words = 25; window size = 12; metric = Euclidian. Fig. 5.7 reveals the influence the resulting number of clusters has, for each of the clustering algorithms used. There is a fairly linear rise in clustering accuracy as the number of clusters increases.



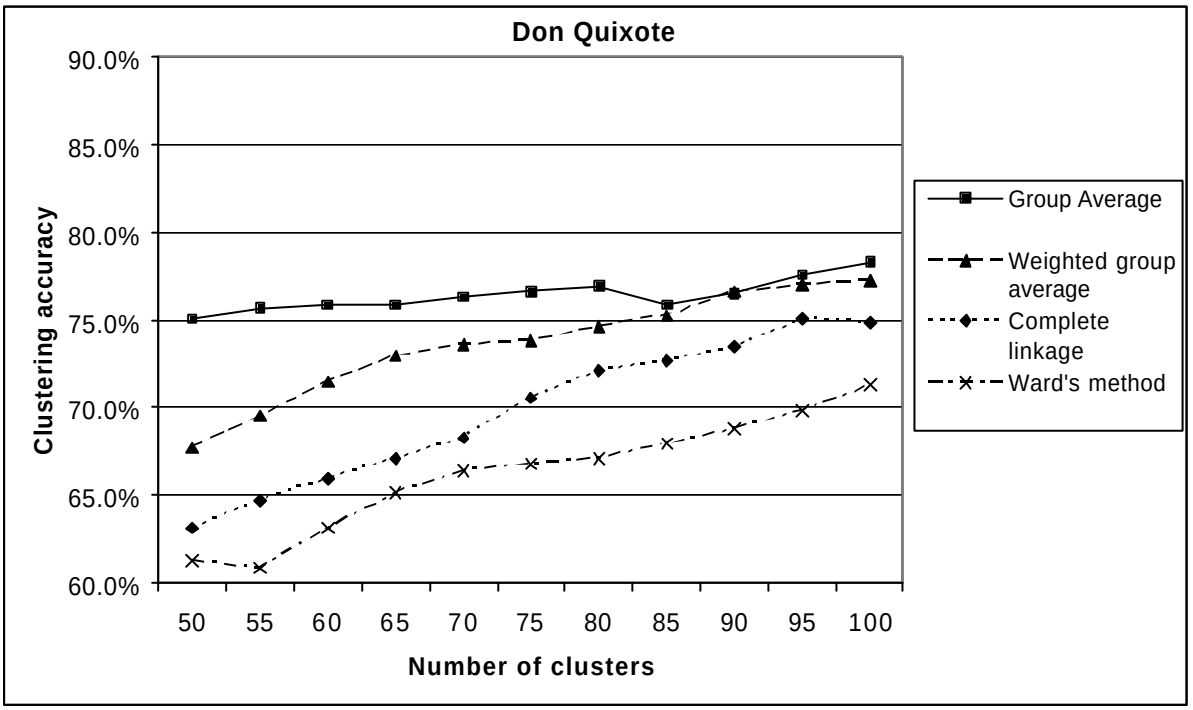


Fig. 5.7 – The relationship between number of clusters and clustering algorithm.

5.7 Semantic Clusters

The results gathered so far are only indicative of how well the clustering has performed on a syntactic level. Judging semantic similarity is not easily quantifiable and is more subjective than with syntax. Nonetheless, many clusters exhibit groups with semantic similarity. Interesting examples from the LOB corpus include:

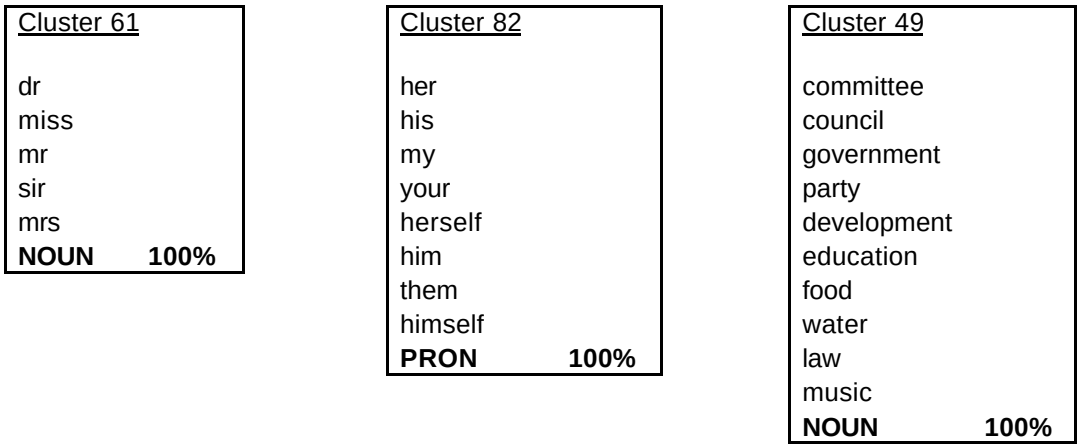


Fig. 5.8 – Examples from the LOB corpus of clusters showing similar semantic properties.

Good examples can also be found in the resulting clusters from the *Don Quixote* corpus.

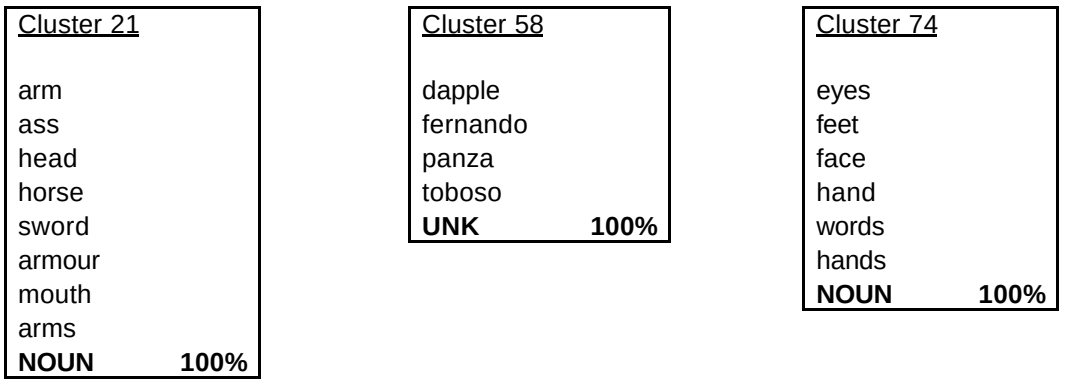


Fig. 5.9 – Examples from the *Don Quixote* corpus of clusters showing similar semantic properties.

Cluster 58 has grouped together proper nouns that feature in the novel.

5.8 Alternative language

Experimentation with the Spanish corpus was not as comprehensive as with the English corpora. Emphasis was purposely placed on understanding how the clustering process worked with English. The motivation for clustering an alternative language is to investigate whether the described method is non-language specific.

The style of experimentation remained as close to the ones performed on the English corpora. The first notable difference was the function words. The words listed in Table 3.1 had to be translated into their Spanish equivalent. This was straightforward in itself, although the translated list was longer due to many words being assigned to a gender.

The most significant difference is the method for evaluating the resulting clusters. A large tagged Spanish corpus comparable in size and detail (in terms of its tagset) to the LOB corpus was not readily available. A tagged Cuban-Spanish corpus was found though it only contained approximately 16,000 words, with 3,000 distinct word tokens. Although only 500 words from the *Don Quixote* corpus were clustered, there were many words that didn't feature in the tagged corpus. Therefore, the tagged corpus was manually supplemented with those missing words, and their parts-of-speech were obtained from *Collins Spanish Dictionary* (Sinclair 1998). The tagset used for the corpus was small, and unfortunately did not differentiate between different types or tenses of verbs.

Tag	**Type of item**
ART	Article
ADJ	Adjective
ADV	Adverb
CCON	Conjunction
EXPL	Interjection
NOUN	Noun
PREP	Preposition
PRON	Pronoun
VERB	Verb

Table 5.5 – The tagset used for the Spanish corpus.

Unsurprisingly, the same properties identified in the English experiments were present in the Spanish experiments. That is, the window size, number of function words and number of resulting clusters improves the clustering accuracy. Group Average once again the highest performing algorithm, and the Manhattan metric consistently outperformed the Euclidian metric for the experiments carried out. These observations are summarised in Table 5.6.

Metric	Clustering Algorithm	Number of Clusters				
		80	85	90	95	100
Euclidian	*Complete Linkage*	74.45%	75.05%	75.70%	76.70%	77.60%
	Group Average	81.15%	81.55%	81.93%	82.00%	82.33%
	Weighted Group Average	79.86%	79.72%	80.91%	80.62%	81.06%
	Ward's Method	73.07%	74.45%	74.46%	74.88%	75.63%
Manhattan	*Complete Linkage*	74.96%	75.73%	76.32%	76.41%	76.23%
	Group Average	84.77%	85.38%	85.38%	85.30%	85.82%
	Weighted Group Average	80.10%	80.37%	81.52%	82.45%	83.31%
	Ward's Method	73.75%	75.16%	75.12%	76.56%	76.87%

Table 5.6 – Summary of results showing the clustering accuracy for the Spanish *Don Quixote* corpus.

5.8.1 Semantic Clustering

There were many promising examples of clusters displaying similar semantic properties. This is encouraging considering the size of the corpus. With the Spanish corpus, it was comparatively common to find clusters containing the same verb in different tenses. For example, cluster 68 lists the verb *to be* in various tenses. Cluster 13 consisted of similar adjectives, and cluster 86 held nouns of parts of the body.

Cluster 13	
aquel	*that (one)*
aquella	*that (one)*
esta	*this*
este	*this*
aquellas	*those*
mi	*my*
su	*his/her*
sus	*his/her*
tu	*your*
mil	*thousand*
ADJ	**100%**

Cluster 86	
manos	*hands*
ojos	*eyes*
pies	*feet*
marido	*husband*
NOUN	**100%**

Cluster 68	
era	
es	
fue	
eran	
son	
eres	
sean	
VERB	**100%**

Fig. 5.10 – Examples of clusters displaying similar semantic properties.

6 Future Work

There is still a great deal that could be done to improve and expand the research undertaken by this project.

6.1 More experiments

The experiments in section 5 would have benefited greatly if each varying factor could have been tested more comprehensively. For example, the number of resulting clusters only ranged from 50 to 100 in increments of 5. It would have been desirable to expand the range and reduce the size of the increment. Not just for the number of clusters, but for all factors being investigated. Also, some of the factors remained relatively static throughout the experimentation. Namely the number of content words to be clustered, but also the type and size of corpora. These factors also need to be understood.

6.2 Automatically obtaining function words

In order to reduce the dependence of expert linguistic knowledge for the entire clustering process, the method of obtaining the function words needs to be revised. This process needs to be automatic. This could simply consist of taking the top *n* ranking words from a given corpus and assuming those words are the function words. This could be rather unpredictable, given that it relies on the type and size of corpus. An example highlighted by Manning and Schütze (2000) showed the frequency counts of the most common words in *Tom Sawyer* by Mark Twain. The 14th most common word was *Tom*. Such a problem is normally overcome by using a large corpus, which contain samples from many different sources. Elliott (2000a and 2000b) has found that function words can be obtained by combining relatively small samples from at least 3 sources: “Three text samples of a few thousand words [are] OK, but four is better to filter out the more common content words.” (Elliott, personal communication)

6.3 Improving the context measure

It is worth exploring the effect of the context measure to see if it can be refined. The context measurement in this project was the distribution of a target content word against the target function words. It would be possible to experiment with assigning weights to certain attributes which are believed to be valuable. For example, the occurrence of a target content word positioned next to a target function word would be more informative than if the occurrence was five words away. As a result, a greater emphasis could be placed on words that appear next to each other. Alternatively, it is worth looking at where they do not occur. Recalling the example distribution displayed in Fig. 3.1, for '*in*' and '*the*'. The fact that *the* never occurred immediately before *in* is equally informative (if not more so) than the fact that *the* occurs most frequent immediately after *in*. Of course, this particular approach becomes less useful for rarer content words; since there are so few occurrences that it would be hasty to attempt any inference.

6.4 Incremental learning

There is clearly more information that can be extracted from the most frequently occurring words. As mentioned in the previous paragraph, certain observations can be made about words which frequently occur together and which ones do not, providing there is enough evidence to support them. These observations could be relied upon as *rules*, which would be much more dependable, than clustering by similarity alone. These rules could be built up incrementally, to overcome the data sparseness problem, for example, using the two most frequent words of a given corpus, e.g., *the* and *of*.:

1. It will be observed that *the* never comes immediately before *of*. As these are the highest ranked words in the corpus, it can be assumed to be a rule.

2. Then, observing the distribution of the 100 most frequent words of the corpus, all words that behave similarly to *of* can be found.

3. If they also never occur immediately after *the*, then the rule can be altered to:
'Words that cluster with *of* never occur immediately after *the*', or 'Prepositions never follow *the*'.

4. If words that cluster with *the* are found to behave similarly, then the amended rule would be: 'Words that cluster with *the* are never followed immediately by words that cluster with *of*', or 'Prepositions never follow determiners'.

By comparing how a word behaves with a given word class, rather than another word, data sparseness is reduced, and should allow more accurate clustering for rarer words.

6.5 Other languages

The use of a second language to investigate the performance of the clustering techniques described in this report returned positive results. It confirmed the theory that clustering algorithms should not be troubled with another language – it is an unsupervised process and does not require prior knowledge of the language to be clustered. However, different languages can vary greatly in their behaviour. Manning and Schütze (2000) mention that the English language lends itself more favourably for inferring part-of-speech from its position, compared to other languages. "In many other languages, word order is much freer, and the surrounding words will contribute much less information about part of speech." Therefore, more thorough experimentation needs to be carried out. Obtaining better resources will be essential. A larger corpus would be beneficial, as too would be a larger tagged corpus for more reliable evaluations.

There is no reason to limit this technique to human languages. Potentially NLL techniques could be applied identify features in other 'language-like' datasets, such as birdsong, signals from dolphins and whales etc. Also there is the potential to search through sequences of DNA to seek patterns automatically.

7 Conclusions

7.1 Review

The aim of the project was to investigate an approach to automatically acquire word classification. After reviewing many methods for tagging, a clustering approach was adopted due to its appropriateness for tasks that need to be *unsupervised*. The context measurement was decided to be the relative distribution of a target content word against the most frequent function words. An algorithm was implemented that could extract this distributional information from an inputted corpus. This data was converted in a vector that represented the distribution of the content word and placed into a high-dimensional vector space. A variety of metrics and clustering algorithms were implemented which could made use of the populated vector space, created a distance matrix which then established which words were similar.

A tool was developed to automatically evaluate the clusters that were produced by the clustering process. Such a tool proved to be essential due to the number of experiments that were performed. For the English language:

Factor	**Number**	**Values**
Number of corpora	2	LOB and Don Quixote
Number of cluster sizes	11	55 – 100 in increments of 5
Number of function words used	6	5 – 30 in increments of 5
Number of window sizes	6	2 – 12 in increments of 2
Number of metrics	2	Euclidian and Manhattan
Number of clustering algorithms[7]	8	Single linkage, Complete linkage, Group average, Weighted group average, Median, Centoid, Centre of gravity, Ward's Method

The total number of experiments = *2* x *11* x *6* x *6* x *2* x *8* = *12,672*.
(The number of Spanish experiments only totalled to 480)

[7] As mentioned in section 3.3.3, eight algorithms were originally implemented, but four performed unsatisfactorily and so not considered in any evaluation for the experiments. However, they still had to be run in order to discover their lack of suitability to this particular problem.

7.2 Experimental findings

With regards to investigation with English corpora:

- Clustering accuracy was higher for the LOB corpus by a mean average of 6.5%. This was expected due to the LOB corpus being over twice the size as *Don Quixote*. Of course, the two corpora are quite different in style, so a direct comparison is not appropriate.
- The best clustering algorithm was found to be Group Average from the automatic evaluation for both corpora.
- The metric had varying success – its effect was dependent on other factors, e.g., the algorithm used, and the corpus too. Overall, Manhattan performed better with the smaller *Don Quixote* corpus, and Euclidian worked better with the LOB corpus. Lund and Burgess (1996) and Hughes (1994) both found Manhattan produced the better results, which fits Shepard's (1980) theory that lower values for *m* in Minkowski's general distance equation, are more suitable for extracting semantic information.
- The clustering performance improved as the number of function words used was increased.
- Clustering accuracy was generally enhanced as the window size increased. This is likely to be due to the fact that the maximum function word separation is eight words, as observed by Elliott (2000a). This means that for any content word, there will always be a function word no greater than ±4 words (equivalent to a window size of eight).
- The number of resulting clusters had a great influence on the effectiveness of the clustering. The greater the number of clusters, the better the performance. As discussed however, it is important to find the right balance between the number of content words being clustered and the resulting number of clusters. There is little information to be gained when the number of clusters is close to either extreme (i.e., very small or close to the number of content words).
- Even with the small *Don Quixote* corpus, the highest accuracy of the clusters managed to reach 84.7%. With the LOB corpus, an accuracy of 87.8%.
- Good evidence of semantic clustering was found for both corpora. Even words such as 'a' and 'an' which could potentially cause trouble (since the words that follow

immediately after are mutually exclusive) were grouped together within the same cluster.

The clustering techniques also showed promising results when the Spanish *Don Quixote* corpus was used. Although the tagged corpus was relatively small, as too was the tagset, it was still adequate for the preliminary studies into the effectiveness of clustering an alternative language. An accuracy of 85.8% was achieved and semantic groups were frequent in the resulting clusters. This shows that the function word profiles were an effective context measure.

The experimental findings indicate that the method proposed by this report is a feasible way for automatically acquiring word classification. A high degree of success was accomplished even with comparatively small corpora. Admittedly, stronger semantic clustering was found in the larger LOB corpus, and can be found in other experiments involving vast corpora (see section 3.4).

Appendix D gives a sample output of the clustering software where:

- corpus: LOB
- clustering algorithm: Complete Linkage
- metric: Manhattan
- number of resulting clusters: 100
- number of function words: 25
- window size: 12

References

(Atwell 1987) Eric Steven Atwell*. A Parsing Expert System which Learns from Corpus Analysis*. In Willem Meijs (ed) - *Corpus Linguistics and Beyond: Proceedings of the ICAME 7th International Conference on English Language Research on Computational Corpora*, pp227-235, Amsterdam, Rodopi. 1987.

(Atwell and Drakos 1987) Eric Steven Atwell and Nikos Drakos*. Pattern Recognition Applied to the Acquisition of a Grammatical Classification System from Unrestricted English Text*. In Bente Maegaard (ed) - *Proceedings of the Third Conference of European Chapter of the Association for Computational Linguistics*, pp56-63, New Jersey, Association for Computational Linguistics. 1987.

(Baker 1975) J.K. Baker*. Stochastic Modelling for Automatic Speech Understanding*. In D.R. Reddy (Ed) - *Speech Recognition*. Academic Press. New York. 1975.

(Baker 1979) J.K. Baker*. Trainable grammars for speech recognition*. In D.H. Klatt and J.J. Wolf (Eds) - *Speech communication papers for the 97th meeting of acoustical society of America*, pp547-550. 1979

(Chomsky 1957) Noam Chomsky. *Syntactic Structures*. The Hague: Mouton. 1957.

(Chomsky 1964) Noam Chomsky. *Current Issues in Linguistic Theory*. The Hague: Mouton. 1964

(Church 1988) K. W. Church. *A stochastic parts program and noun phrase parser for unrestricted text*. In *Second Conference on Applied Natural Language Processing*, pp. 136-143. ACL. 1988.

(Cutting *et al*. 1992) D. Cutting, J. Kupiec, J. O. Pedersen and P. Sibun. *A Practical part-of-speech tagger*. In *Third Conference on Applied Natural Language Processing*, pp. 133-140. ACL. 1992.

(Dempster *et al*. 1977) A. P. Dempster, N. M. Laird and D. B. Rubin. *Maximum likelihood from incomplete data via the EM algorithm*. Journal of the Royal Statistical Society, 39(1), 1-21. 1977.

(Elliott *et al*. 2000a) J. Elliott, E. Atwell and W. Whyte. *Increasing Our Ignorance of Language: Identifying language Structure In An Unknown Signal*. In*: Proceedings of 4th International Conference on Computational Natural Language Learning*, pp 25-30. Association of Computational Linguistics. New Jersey. 2000.

(Elliott *et al*. 2000b) J. Elliott, E. Atwell and W. Whyte. *Language identification in unknown signals*. In *Proceeding of COLING'2000, 18th International Conference on Computational Linguistics*, pp1021-1026, Association for Computational Linguistics (ACL) and Morgan Kaufmann Publishers, San Francisco. ISBN: 1-55860-717-X (2 volumes). 2000.

(Everitt 1993) B. Everitt. *Cluster Analysis (3rd Edition)*. Edward Arnold, London. 1993.

(Finch and Chater 1992) S. Finch and N. Chater. *Bootstrapping syntactic categories*. In *Proceedings of the 14th Annual Meeting of the Cognitive Science Society*, pp 820-825. Hillsdale, New Jersey. 1992.

(Francis 1979) W. N. Francis. *A tagged corpus – problems and prospects*. In S. Greenbaum, G. Leech and J. Svartvik (Eds). *Studies in English linguistic for Randolph Quirk*, pp 192-209. Longman, London and New York. 1979.

(Francis and Kucera 1982) W. N. Francis and H. Kucera. *Frequency Analysis of English Usage*. Houghton Mifflin, Boston. 1982.

(Francis and Kucera 1989) W. N. Francis and H. Kucera *Manual of Information to Accompany a Standard Corpus of Present-Day Edited American English, for use with Digital Computers (Corrected and Revised edition)*. Department of Linguistics, Brown University, Providence, Rhode Island. 1989.

(Garside 1987) R. Garside. *The CLAWS word tagging system*. In R. Garside, G. Leech and G. Sampson (Eds), *The Computational Analysis of English*, pp. 30-41. Longman, London. 1987.

(Garside *et al*. 1997) R. Garside, G. Leech and A. McEnery. *Corpus Annotation*. Longman, London and New York. 1997.

(Greene and Rubin 1971) B. B. Greene and G. M. Rubin. *Automatic grammatical tagging of English*. Department of Linguistics, Brown University, Providence, Rhode Island. 1971.

(Harris 1962) S. Z. Harris. *String Analysis of Sentence Structure*. Mouton, The Hague. 1962.

(Haslerud and Stenström 1995) V. Haslerud and Anna-Brita Stenström. *The Bergen Corpus of London Teenager Language (COLT)*. In G. Leech, G. Myers & J. Thomas (eds.). *Spoken English on computer.* London: Longman, 235–242. 1995.

(Hughes 1994) John Hughes. *Automatically Acquiring a Classification of Words*. PhD Thesis. School of Computer Studies, University of Leeds. 1994.

(Johansson *et al*. 1986) S. J. Johansson, E. S. Atwell, R. Garside and G. Leech. *The tagged LOB Corpus. Users' manual*. The Norwegian Centre for the Humanities, Bergen. 1986.

(Jurafsky and Martin 2000) Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall. New Jersey. 2000.

(Kiss 1972) G. R. Kiss. *Grammatical Word Classes: A Learning Process and its Simulation.* Psychology of Learning and Motivation. 7. pp1-41. 1972.

(Klein and Simmons 1963) S. Klein and R. F. Simmons. *A computational approach to grammatical coding of English words*. Journal of the Association for Computing Machinery, 10(3), 334-347. 1963.

(Kucera and Francis 1967) H. Kucera and W.H. Francis. *Computational analysis of present-day American English*. Brown University Press, Providence, Rhode Island. 1967.

(Lance and Williams 1967) G.N. Lance and W.T. Willams. *A General theory of classification sorting strategies*. 1. Hierarchical Systems, Comp. J., 9, pp373-380. 1967

(Leech 1993) G. Leech. *100 Million Words of English: The British National Corpus (BNC) Project.* English Today. 1993.

(Leech *et al*. 1994) G. Leech, R. Garside and M. Bryant. Claws4: *The tagging of the British National Corpus*. In *COLING-94*, Kyoto, pp. 622-628. 1994.

(Levy and Bullinaria 2001) J.P. Levy and J.A. Bullinaria. *Learning Lexical Properties from Word Usage Patterns: Which Context Words Should be Used?* In: R.F. French & J.P. Sougne (Eds) - *Connectionist Models of Learning, Development and Evolution: Proceedings of the Sixth Neural Computation and Psychology Workshop,* 273-282. London: Springer. 2001.

(Lund and Burgess 1996) K. Lund and C. Burgess, *Producing high-dimensional semantic spaces from lexical cooccurrence*, Behavior Research Method, Instruments, & Computers 28(2), pp 203-208. 1996

(Manning and Schütze 2000) C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press. Cambridge, Massachusetts. 2000.

(Marcus *et al*. 1993) M. P. Marcus, B. Santorini and M. A. Marcinkiewicz. *Building a large annotated corpus of English: The Penn treebank*. Computational Linguistics, 19(2), 313-330. 1993.

(Marshall 1983) I. Marshall. *Choice of grammatical word-class without global syntactic analysis: Tagging words in the LOB corpus*. Computers and the Humanities, 17, 139-150. 1983.

(Merialdo 1994) B. Merialdo. *Tagging English text with a probabilistic model*. Computational Linguistics, 20(2), 155-172. 1994.

(Oakes 1998) M. P. Oakes. *Statistics for Corpus Linguistics*. Edinburgh University Press. 1998.

(Ormsby 1885) Miguel De Cervantes. *Don Quixote de la Mancha*. 1665. Trans. John Ormsby. 1885.

(Redington *et al*. 1998) Martin Redington, Nick Chater and Steven Finch. *Distributional Information: A Powerful Cue for Acquiring Syntactic Categories*. Cognitive Science, Vol 22 (4), pp 425-469. Cognitive Science Society. 1998

(Shepard 1980) Roger N. Shepard. *Multidimensional scaling, tree-fitting, and clustering.* Science, 210, pp390-398. 1980.

(Simpson and Weiner 1989) J. A. Simpson and E. S. C. Weiner (ed). *Oxford English Dictionary*. 2nd ed. Oxford: Clarendon Press, 1989

(Sinclair 1987) J. Sinclair. *Looking Up: An Account of the COBUILD Project in Lexical Computing*. Collins, Glasgow. 1987.

(Sinclair 1998) L. Sinclair (ed). Collins Spanish Dictionary, Plus Grammar. HarperCollins, Glasgow. 1998.

(Stolz *et al*. 1965) W. S Stolz, P. H. . Tannenbaum and F. V. Carstensen. *A Stochastic approach to the grammatical coding of English*. Communications of the ACM, 8(6), 399-405. 1965.

(Svartvik 1990) Jan Svartvik (ed). *The London-Lund Corpus of Spoken English: Description and Research*. Lund University Press, Lund, Sweden. 1990.

(Taylor and Knowles 1988) L.J. Taylor and G Knowles. *Manual of Information to Accompany the SEC Corpus: The machine readable corpus of spoken English*. University of Lancaster. 1988.

(Ward 1963) J. H. Ward. *Hierarchical Grouping to Optimize an Objective Function*. Springer-Verlag, Berlin. 1963.

(Zipf 1949) G.K. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison Wesley Press. New York. 1949.

(Zupan 1982) Jure Zupan. *Clustering of Large Data Sets*. John Wiley and Sons, Chichester. 1982.

Appendix A – Reflection on project experience

The final year project was like no other piece of assessed work required during my computing degree. No prior coursework was of a comparable scale of effort. As a result, I was very much unprepared when approaching my project.

My first piece of advice for any student is to select a project that you are actually interested in. Of course, the project is a compulsory component of the degree, however, if you can make it more than just a project done for the sake of it, then you will benefit in the long run. I was excited when I began my research. As a result, I have enjoyed my area of study so much that I have altered my career plans in order to stay at the university to do a PhD! It is an excellent opportunity to study a topic that *you* want.

Your project supervisor will be the most important person in ensuring success with your project. Absorb as much advice as you can. Yet, students should feel free to seek advice from other members of staff within the department. For projects in the field of NLP, resources are not always easy to obtain, e.g., tagged corpora are not freely available. Therefore, maintain a good rapport with the academics who can assist you in acquiring resources you may need.

For any research project, it is easy to carried away since there is often no limit to what you can investigate for a given topic. Therefore, focus on the core objectives only. Try to define the scope of your project to ensure you do not start going off on a tangent. My personal experience was that I took on more than was required. I experimented with too many factors when I should have honed in on a small selection and performed comprehensive tests on just those. It is a difficult thing to gauge since you are unlikely to have undertaken a project this large. Therefore, seek advice from your supervisor regarding this issue.

A comprehensive literature review is essential for whatever topic you research. However, it may be worthwhile to perform a software review. A number of tools I personally implemented for my project were re-engineering software already freely available. I simply was not aware of their existence at the time. Thus, I sacrificed a great deal of time writing code which could have been better spent on other areas of the project. The three rules for a successful project are: 1) time management 2) time management and 3) time management!

Appendix B – Discussion paper by Bill Whyte and John Elliott

I am proposing a method which I hope will firstly identify a set of 'word classes' broadly equivalent to parts of speech, and secondly, act as the beginnings of a novel part-of-speech tagger. The initial motivation is as a way of detecting structural features in completely unknown language-like strings.

We have a table with rows headed by function words, typically 10-30 (only 5 shown for simplicity)

Fn words -> Content wrds | v	Fn1	Fn2	Fn3	Fn4	Fn5

and a largish number of common, content words, selected from the test corpora, listed in the first column.

We would like to able to (but probably can't without a bit more fiddling - see later) create a row of numbers, each of value from 1-4 (say) for each content word. Each number represents the separation between the content word and a function word.
(NOTE: we need to do this for the cases where the content word precedes the function word and where it follows it, but I'm keeping it simple for the time being).
So, we can define and enumerate a 'relation' between content word PARROT and the set of function words {Fi} e.g. PARROT{ the, of, a, in …..} = {1,3,1,2…….}

OUR BIG HYPOTHESIS: is that if we can calculate enough ***complete*** relations for enough words, then we'll have got enough examples for every important class of words (i.e. parts of speech, sort of) AND we shall see that the relations for individual words will cluster into distinct groups AND it will be possible to use simple clustering techniques which do not rely on the complex statistics of word position.

A SIMPLIFICATION: for various reasons, I suggest we try to 'collapse' the behaviour of relations for individual words: the relation for content word 'parrot' with respect to function word 'the' in the phrase 'the green parrot' has value 2 ('green' comes in between them.) but value 1, in the phrase 'the parrot'. The recommendation is, 'for any relation between a word and a specific fn word, use the smallest value that occurs for any instance of that word. This would seem reasonable, will reduce data processing and I think it will actually help with the cluster analysis. Note that we are thereby using 'closest approach to Fi' as a discriminating feature.

IF EVERYTHING GOES WELL (but it won't): then we end up with a reasonable number, say a few hundred (it needs to that for statistical accuracy), complete relations, i.e. a complete set of values for the minimum separation between each of our content words and every one of the function words.

I'm quite hopeful, in this case, that we will have a number of relations that have the same values in them and that the words they relate to will quite reasonably be describable as belonging to the same class. I even reckon there is a good chance that 'class' may be roughly equivalent to 'part of speech' in the trad sense. So, simply choose the first half dozen or so distinct relationships as being word class discriminators.

THE BIG PROBLEM: I see one particularly large problem (plus some others, e.g. polysemy, which I won't go into) - quite simply, we won't be able to get a complete set of relation values. Remember, we need to seek out lots of occurrences of a lot of content words where each of them is within 3 or 4 of each function word. (If they ever are). AND, if we don't get the minimal values for each of these distances, then our data is corrupted.

A POSSIBLE APPROACH TO A SOLUTION: firstly, remember that it doesn't matter which content words we use specifically. As long as we have enough text, we can hunt around for possible solutions for any one word but give up if it's looking bleak and simply choose another one. Secondly, a more complex approach might be to tentatively decide that two words are likely to belong to the same word-class, on the basis of a shared similarity between their (incomplete) relations and some additional tests (see next para) and allow each one to inherit the missing bits from each other.
So, how do we 'confirm' one of these tentative equivalences? Let's remember that, for any one content word, we will probably have acquired a number of non-closest approaches (the GREEN parrot, the VERY LARGE parrot etc). We may also have acquired similar for the other content word that we think might be of the same class, eg. 'the RED dog', 'the LARGE dog' etc. Suppose, instead of throwing these examples away in favour of minimals 'the parrot', 'the dog', we keep them stored. Now, suppose we tentatively associate 'parrot' and 'dog' in the same word class, on the basis of part of their relations being the same, for example they both have {… ….the…..} = {……1……}. [In practice, we'd need more similarity]. We now further test the probability of the association by looking at the stored data :

Fn words -> Content wrds | v	Fn1	Fn2	THE	Fn4	Fn5
DOG			Minimal =1 Non-mins: Red Large		
PARROT			Minimal =1 Non-mins: Green Very large		

We see they both have 'large' at distance 1. Therefore, it's looking a bit more probable that they are of same class. We can do more than this. Suppose we explore the relations for 'red' and for 'green' wrt the function words. If we get a reasonably high correlation, even on partial relations, then this is further evidence.

RULES AND HEURISTICS: we see that the algorithm has a clearly defined part, that is the getting of 'easy' minimals directly and a less clearly defined part which is going to involve a bit of trial and error including some heuristics for back-tracking when the approaches in the previous para conflict on their categorisation of a specific word. One hopeful point is that we don't have to process every word to its end. We are only after a sufficient number and range of successful words. Therefore, we may be able simply to throw away examples that are looking doubtful or ambiguous.

COMPUTATIONAL ASPECTS

Suggest that the corpora are stored in a couple of ways that might speed up computation significantly. First of all, a simple two dimensional list of word against left-right, top down order. E.g.

If total corpus is Here is a dog. It is called spot, then list is simply:

1	2	3	4	5	6	7	8
here	is	a	dog	its	name	is	Spot

(In practice, we need to think about punctuation).

Secondly, index the words alphabetically and against serial position:

A	dog	here	is	its	name	Spot
3	4	1	2,7	5	6	8

An additional list with the function words in this way might also be useful.

If this indexing is done before the rest of the processing, it may make it much quicker to find repeated occurrences and to allow backtracking.

Appendix C – The LOB corpus tagset

Tag	Description	Examples
&FO	formula	10*:-1**: dE *:238**:U a*;n**; T*:-3/2**: E*;p**;(P) R*?8r(cdE.cde) ... [See note 1]
&FW	foreign word	de Welt von Retour Flamme route Musique Ancienne Pro unheimliche Opus baraka Biennale Internationale Novum sine die cantabile letzt bru"cke ...
!	exclamation mark	!
(	opening parenthesis	(
)	closing parenthesis	)
***'**	opening quotation mark	*' *" [See note 2]
****'**	closing quotation mark	**' **" [See note 2]
***-**	dash	*- [See note 2]
,	comma	,
.	full stop	.
...	ellipsis	...
:	colon	:
;	semicolon	;
?	question mark	?
ABL	determiner/pronoun, pre-qualifier	such quite rather such-and-such
ABN	determiner/pronoun, pre-quantifier	all half
ABX	determiner/pronoun, double conjunction or pre-quantifier	both
AP	determiner/pronoun, post-determiner	more most last several next own other many much same less former only very few fewer latter least overmuch ain kast
AP"	determiner/pronoun, post-determiner, ditto	few good many little [See note 3]
AP$	determiner/pronoun, post-determiner, genitive	latter's former's other's
APS	determiner/pronoun, post-determiner, plural	others
APS$	determiner/pronoun, post-determiner, plural, genitive	others'
AT	article, singular	a an every
ATI	article, singular or plural	the no nae ye zee de ze
BE	verb "to be", infinitive or imperitive	be
BED	verb "to be", past tense, 2nd person singular or all persons plural	were
BEDZ	verb "to be", past tense, 1st and 3rd person singular	was
BEG	verb "to be", present participle or gerund	being
BEM	verb "to be", present tense, 1st person singular	am 'm
BEN	verb "to be", past participle	been
BER	verb "to be", present tense, 2nd person singular or all persons plural	are 're art 'rt ai
BEZ	verb "to be", present tense, 3rd person singular	is 's iss ees ai
CC	conjunction, coordinating	and but or nor as & yet / 'n and/or an' only n'
CC"	conjunction, coordinating,	well as

	ditto	
CD	numeral, cardinal	1958 13 two 280,000 20 1959 28.5 400 eight 2 1949 six seven ten 1,400 16 9.40 five 100 four fifty 89 5.30 287 million 1/2 2.35 forty nine 6.55 ...
CD$	numeral, cardinal, genitive	8's 3's 5's 4's
CD-CD	numeral, cardinal, hyphenated pair	1955-6 15-20 1861-1940 1-6 2-0 3-1 33-1 12-1 300-400 0-3 10,000-15,000 1611-1961 six-five 51.1-3 280-338/39 ...
CD1	numeral, cardinal, one	one 1 'un
CD1$	numeral, cardinal, one, genitive	one's 1's
CD1S	numeral, cardinal, one, plural	ones 'uns
CDS	numeral, cardinal, plural	hundreds thousands dozens fifties two-thirds millions 1830's '30s forties middle-thirties sevens '20s 30's 1750s 'forties nines five-sixths zeros ...
CS	conjunction, subordinating	though that as while if because before than since whether for once except until provided unless although even lest now till such so but albeit whereas in considering nisi like whilst 'n whereupon save altho' tho' directly 'cos 'cause immediately
CS"	conjunction, subordinating, ditto	if that as though so far order
DO	verb "to do", uninflected present tense, infinitive or imperitive	do
DOD	verb "to do", past tense	did
DOZ	verb "to do", present tense, 3rd person singular	does doth
DT	determiner/pronoun, singular	another this that each zis zat anudder
DT$	determiner/pronoun, singular, genitive	another's
DTI	determiner/pronoun, singular or plural	any some enough
DTS	determiner/pronoun, plural	these those
DTX	determiner, pronoun or double conjuction	either neither
EX	existential there	there
HV	verb "to have", uninflected present tense, infinitive or imperitive	have 've hast of 'ave
HVD	verb "to have, past tense	had 'd
HVG	verb "to have", present participle or gerund	having havin'
HVN	verb "to have", past participle	had
HVZ	verb "to have", present tense, 3rd person singular	has 's hath ai
IN	preposition	by from at of on for into since in to with despite round as about over without towards behind during under beyond after because against outside including among like apart ...
IN"	preposition, ditto	of from spite with to front as for means opposed top between against la regards board versus
JJ	adjective	large likely out-dated adequate nationalist federal united national elected colonial full proposed secret central final unsatisfactory gross unconstitutional angry human heavy hostile economic monstrous warm-hearted ...
JJ"	adjective, ditto	up off luxe round cut weight priori hoc vires lived board fashioned

JJB	adjective, attributive-only	left-wing rival chief overall main once-and-for-all prime past nuclear-disarming anti-apartheid American-born built-in 89-year-old joint pro-communist centre second-row top ...
JJB"	adjective, attributive-only, ditto	army called
JJR	adjective, comparative	higher better worse easier wider tougher lesser nicer fairer worthier prettier neater noisier deeper happier nobler nearer slower bolder shallower faster-moving ...
JJR"	adjective, comparative, ditto	wearing
JJT	adjective, superlative	best fiercest bitterest largest toughest thorniest rarest humblest freshest sweetest clearest best-regulated kindest simplest-of-all handsomest strangest tallest ...
JJT"	adjective, superlative	selling
JNP	adjective, word-initial capital	African British Rhodesian anti-Negro German Manchu-Edwardian Yugoslav Asian Congolese inter-African Nazi Olympic Ritzy Anglo-American Persian Elizabethan Teutonic un-Italian Marxist Ritzy Norman Viking Luddite Presbyterian Churchillian Orwellian Kentish ...
MD	modal auxillary	may will should would can must might could need 'll shall 'd ought wilt mayest dared maun cou'd dare shoud shoulda 'ud wikk
NC	cited word	many thanks Jimmy ret-key s nonsense Directors' emoluments always only high decadent mouse sous Gita Ghita explode ...
NN	noun, singular, common	life move bill existence institution sentiment abolition independence association bureau investigation service post present spot sum drain answer rejection blow taxation ...
NN"	noun, singular, common, ditto	mortem blanche d'oeuvre douloureux garde d'hotel how up obscura hoccery d'affaires pectoris grata ego between
NN$	noun, singular, common, genitive	protectorate's labour's doctor's parliament's man's child's hour's airliner's week's conference's regiment's unit's pilot's orchestra's river's library's ...
NNP	noun, singular, common, word-initial capital	Chinese Irishman American Australian Negress English Scottish Briton Vansittartism Sinhalese Jew Whitgiftian Berliner Gaitskellism Jesuit Lancastrian Belgian Augustinianism Amorite Anabaptist Druze Celt Rugbeian Highlander ...
NNP$	noun, singular, common, word-initial capital, genitive	Englishman's Russian's Greek's Hungarian's Eskimo's Genoese's Turk's Frenchman's Prussian's Corsican's Canadian's ...
NNPS	noun, plural, common, word-initial capital	Africans Americans French Germans Nazis Arabs Anglo-Saxons Scandinavians Romans Pan-Somalis Berliners Ceylonese Cestrians Wearsiders Czechs Hessians Victoriana Dalmatians Brownists Bantu Kelts Slavs Medes Hindoos ...
NNPS$	noun, plural, common, word-initial capital, genitive	Germans' Tunisians' Americans' Africans' Nyasalanders' Spaniards'
NNS	noun, plural, common	peers nominees steps plans discussions organisations drugs conditions opponents details cuts changes foods families deeds words supplies measures police cars demonstrators ...
NNS"	noun, plural, common, ditto	d'appui ups
NNS$	noun, plural, common, genitive	settlers' neutrals' years' pro-communists' nightingales' players' footballers' sportsmen's officers' women's staffs' contributors' employers' hairdressers' breeders' ...
NNU	noun, abbreviated unit of measurement	\0s \0min \0mph \0d \0in \0p.c \0lb *+1,755million *+900,000 *+3,607,000 *+2 *+720 ... [See note 4]
NNU"	noun, abbreviated unit of measurement, ditto	\0cent cent \0yd [See note 4]

NNUS	noun, abbreviated unit of measurement, plural	\0pts \0yds \0gns *+s \0pp \0mins \0hrs \0revs \0galls \0lbs \0ins [See note 4]
NP	noun, singular, proper	Trevor Williams Michael Manchester Foot-Griffiths Bell Karen Roy Dennis Welensky Rhodesia Nkumbula Macleod Julius Accra Ellender Adenauer George Enoch France Corell-Barnes Selwyn ...
NP$	noun, singular, proper, genitive	Cheung's Griffith's Oxford's England's Guy's Swansea's Conroy's Zealand's Kent's London's Reid's Margaret's Windsor's Chatterley's Nancy's Sibelius's Shakespeare's Khruschev's ...
NPL	noun, singular, locative, word-initial capital	House Sea Hotel Airport Square Plain Island Palace Loch Cape Town River Gallery Yard Cove Park University Parade Mount Head Fountain Colliery Shipyard Citadel ...
NPL$	noun, singular, locative, word-initial capital, genitive	City's Garden's Moor's Theatre's Church's Marsh's College's
NPLS	noun, plural, locative, word-initial capital	Plains Locks Cottages Hills Colleges Universities Schools Churches Sands Galleries Marshes Downs Grottos Islands Isles Farms Pikes Roads Straits Broads Mountains Steps Levels Meadows Precincts Fields Counties Halls Buildings Gardens Galeries Woods Fens Towers Prisons Banks Moors Villages
NPLS$	noun, plural, locative, word-initial capital, genitive	Universities'
NPS	noun, plural, proper	Maritimes Wolves Barbarians Wasps Alps Penguins Brittains Debenhams Beechams Bents Mitchells Courtaulds Salems Spurs Wileys Balkans Lennons Greyfaces Tele-Bins Loyals ...
NPS$	noun, plural, proper, genitive	Bents' Cortaulds' Spurs' Wolves' Josephs' Rovers' Mudlarks' Beddises' Loyals' Shadows' Merwes' Caxtons' Marshams' Barkers' Frys' Slaytons' Stevens' Apaches' Pentlands' Cadwells' Swansons' Robertsons'
NPT	noun, singular, titular, word-initial capital	\0Mr \0MP Sir Premier Secretary Prime Minister President Senator \0Dr Prince \0St \0Pres Professor Herr \0Mrs \0C.I.G.S Rector \0P.C \0Hon \0Det.-Con \0D.C \0D \0PC \0Chief-Insp ... [See note 4]
NPT"	noun, singular, titular, word-initial capital, ditto	\0P
NPT$	noun, singular, titular, word-initial capital, genitive	President's Minister's Premier's Queen's Princess's Duke's King's Ambassador's Earl's Chancellor's Regent's Director's Lord's Pope's Registrar's Emperor's Lady's Laird's Vicar's Commander's Captain's Ma's Rector's Prince's General's ...
NPTS	noun, plural, titular, word-initial capital	\0MPs Lords Chiefs Ministers Comrades \0M.P.s Premiers Deans Representatives Saints Presidents Commandants Sons \0Messrs \0Clrs Mayors Aldermen Ambassadors Directors \0M.P.'s Tsars Emperors \0C.O.'s Rabbis Masters Knights Kings Brothers ...
NPTS$	noun, plural, titular, word-initial capital, genitive	Speakers' \0MPs' Sons' Directors'
NR	noun, singular, adverbial	tomorrow today yesterday south February Monday home July tonight Sunday north October September December January west east March \0Nov Wednesday to-night south-east Tuesday August Saturday May June to-day \0Feb April north-west \0W ...
NR$	noun, singular, adverbial, genitive	today's yesterday's to-night's Wednesday's to-morrow's west's Sunday's Saturday's tonight's Monday's tomorrow's home's
NRS	noun, plural, adverbial	homes Saturdays Tuesdays Mondays Sundays Fridays Thursdays
NRS$	noun, plural, adverbial,	[See note 5]

	genitive	
OD	numeral, ordinal	second first third thirty-ninth fourth sixth seventh 75th 19th 3rd 4th 6th 2nd twentieth 1,000th 44th fifteenth ...
OD$	numeral, ordinal, genitive	[See note 5]
PN	pronoun, nominal	so anybody nothing no anyone none everything something anything nobody someone everybody somebody no-one some nuffin' ought somethin' nothin' summat nossings somep'n
PN"	pronoun, nominal, ditto	one
PN$	pronoun, nominal, genitive	everyone's everybody's anybody's something's anyone's no someone's
PN$"	pronoun, nominal, genitive, ditto	one's
PP$	determiner, possessive	his their my our its her your thy thine tha 'is yer me
PP$$	pronoun, possessive	ours mine theirs yours his hers thine
PP1A	pronoun, personal, nominative, 1st person singular	I
PP1AS	pronoun, personal, nominative, 1st person plural	we wee
PP1O	pronoun, personal, accusative, 1st person singular	me
PP1OS	pronoun, personal, accusative, 1st person plural	us 's
PP2	pronoun, personal, nominative or accusative, 2nd person	you ye thee thou y' ya tha yuh
PP3	pronoun, personal, nominative or accusative, 3rd person singular	it 't
PP3A	pronoun, personal, nominative, 3rd person singular	he she 'e
PP3AS	pronoun, personal, nominative, 3rd person plural	they
PP3O	pronoun, personal, accusative, 3rd person singular	him her 'er 'im
PP3OS	pronoun, personal, accusative, 3rd person plural	them 'em
PPL	pronoun, singular, reflexive	himself itself myself herself yourself oneself ourself thyself
PPLS	pronoun, plural, reflexive	themselves ourselves each one yourselves one-another
PPLS"	pronoun, plural, reflexive, ditto	other another another's other's
QL	qualifier, pre	too least most very as so more that less mighty real awfully stark this sound precious
QLP	qualifier, post	enough indeed
RB	adverb	forward still together violently once immediately bluntly clearly long obviously somewhere too truly seriously accurately profoundly rapidly superbly about ever entirely overseas ...
RB"	adverb, ditto	and large right particular least course once little last short again than well general from full long so on common certain alia the less main facto first yet se brief but ...
RB$	adverb, genitive	else's

RBR	adverb, comparative	later longer earlier more less better faster worse sooner deeper higher harder closer nearer farther cheaper heavier lower poorer oftener slower louder quicker ...
RBT	adverb, superlative	best most least fastest lowest worst nearest farthest closest longest furthest
RI	preposition, adverbial, lacking compliment	before within between above since after with near against alongside opposite but without below besides beyond beneath underneath like
RN	adverb, nominal	now then here there downstairs upstairs indoors tho inland here-and-now down-town zen hyar downtown
RP	adverb, particle	down up in through on off out apart about back over round away outside across around aboard inside aside by past behind under forth to oot
TO	infinitival to	to so in
TO"	infinitival to, ditto	as to order
UH	interjection	yes please well \0O.K oh no aw goodbye ah gee whiz sure hey presto wham amen say why dear good-morning hurrah aye welcome boy oi bang er hi hullo goddammit ...
VB	verb, base: uninflected present, imperitive or infinitive	stop gather turn abolish put take appear prop favour drop meet discuss want fall give consult delay sit attend pass pay help try mention point say increase run know ...
VB"	verb, base: uninflected present, imperitive or infinitive; ditto	pedal stitch shop
VBD	verb, past tense	opposed brought said telephoned went denied told remained added renewed arose wanted meant talked flew covered noted hoped thanked delivered supported felt noticed agreed ...
VBG	verb, present participle or gerund	replying addressing changing switching recommending hoping preserving provoking preventing wearing working winding accepting recalling ordering listening using ...
VBN	verb, past participle	made put backed abolished created recommended transmitted jailed decided presented cancelled prepared discussed used posted regarded written indicated favoured ...
VBZ	verb, present tense, 3rd person singular	believes remains gives smears shocks cables says needs goes seems professes reports becomes publishes hopes attacks cracks feels regards claims suggests comes speaks outlines ...
WDT	WH-determiner, interrogative	what whatsoever whatever which whichsoever whichever vich
WDT"	WH-determiner, interrogative, ditto	ever
WDTR	WH-determiner, relative	which
WP	WH-pronoun, interrogative, nominative or accusative	who whoever wot
WP$	WH-pronoun, interrogative, genitive	whose
WP$R	WH-pronoun, relative, genitive	whose
WPA	WH-pronoun, nominative	whosoever
WPO	WH-pronoun, interrogative, accusative	whom-ever whom
WPOR	WH-pronoun, relative, accusative	whom
WPR	WH-pronoun, relative, nominative or accusative	who that
WRB	WH-adverb	when wherever where how why however whenever wherein whereby whence whereof whereunto whereon

XNOT	negator	not n't na
ZZ	letter of the alphabet	G-91 F B zh2014 A T-34 bf alp P A20 X D pi O F11309 a b Q M R4 x z q y H M1 J1 M2 J2 n S U c e d f ...

Notes

1. Formulas contain many symbols that were represented by special character sequences in the pre-formatting stage before the tagger was applied to the LOB corpus. This explains the strange appearance of these examples of formulas taken directly from the corpus.
2. The LOB corpus tagger assumed that the input had been pre-formatted. Mostly the text was left as it would appear on a typed page but some characters where represented differently. Amongst these were the quote character and the dash character. The AMALGAM version of the LOB tagger will recognise and annotate these characters in their usual forms: ` ' " -
3. Ditto tags were applied to words whose role changes from their normal syntax when applied in cerain combinations. The first word of the combination is tagged as normal and all subsequent words are given the first word's tag plus the ditto symbol ("). For example, the combination "so as to" is tagged TO TO" TO".
4. Abbreviations in LOB were signalled by adding '\0' to the start of the abbreviated token. The AMALGAM version of the LOB tagger will handle abbreviations whether or not they have the '\0' prefix.
5. The tag classes NRS$ and OD$ were designed by the LOB corpus developers but have no examples because there weren't any in the LOB corpus itself. NRS$ could be used to tag a word like Sundays' and OD$ the word third's.

Appendix D – Clustering output

Cluster 1
1
2
3
4
5
10
6
b
c
others
women
CARD 63.64%

Cluster 2
a
an
any
every
our
the
almost
ever
i
you
per
ART 36.36%

Cluster 3
able
going
ADJ 50%

Cluster 4
about
at
in
on
over
than
to
above
against
under
within
across
towards
PREP 100%

Cluster 5
act
age
school
war
class
section
countries
NOUN 100%

Cluster 6
added
asked
told
came
went
PAST 100%

Cluster 7
after
as
for
with
without
half
death
mind
again
here
now
then
once
everything
ADV 35.71%

Cluster 8
ago
however
perhaps
since
while
ADV 60%

Cluster 9
air
world
land
case
way
cases
NOUN 100%

Cluster 10
all
another
many
one
some
such
ten
ART 42.86%

Cluster 11
alone
away
down
off
together
along
around
round
PREP 87.50%

Cluster 12
already
also
not
still
usually
ADV 80%

Cluster 13
although
if
when
whether
whom
SCON 60%

Cluster 14
always
often
probably
sometimes
ADV 100%

Cluster 15
am
m
ve
PRES 66.67%

Cluster 16
american
local
modern
political
social
ADJ 100%

Cluster 17
among
during
near
and
or
which
outside
PREP 57.14%

Cluster 18
anything
me
us
PRON 100%

Cluster 19
are
were
became
having
making
taking
he
she
they
we
who
PRES 36.36%

Cluster 20
area
field
period
state
meeting
point
stage
NOUN 100%

Cluster 21
art
life
power
business
children
men
people
things
course
NOUN 100%

Cluster 22
back
out
up
open
d
said
thought
got
just
like
s
ADV 36.36%

Cluster 23
be
been
being
less
more
no
too
re
certainly
therefore
thus
ADV 63.64%

Cluster 24
because
until
yes
SCON 66.67%

Cluster 25
become
get
make
take
PRES 100%

Cluster 26
before
but
that
though
yet
nor
CCON 50%

Cluster 27
began
want
wanted
PAST 66.67%

Cluster 28
behind
by
from
into
through
upon
heard
read
PREP 75%

Cluster 29
believe
feel
think
PRES 100%

Cluster 30
best
most
total
company
family
words
effect
problem
question
NOUN 77.78%

Cluster 31
better
right
even
only
rather
gone
four
two
six
free
love
ADJ 27.27%

Cluster 32
between
of
where
called
left
reached
PAST 50%

Cluster 33
black
car
church
house
town
eyes
face
head
voice
NOUN 88.89%

Cluster 34
body
name
book
word
minister
NOUN 100%

Cluster 35
both
its
their
these
this
itself
themselves
DET 57.14%

Cluster 36
boy
girl
man
morning
NOUN 100%

Cluster 37
britain
london
order
NOUN 100%

Cluster 38
british
national
other
west
city
country
future
past
NOUN 75%

Cluster 39
brought
had
has
have
is
was
due
PAST 42.86%

Cluster 40
building
labour
service
society
trade
common
general
public
french
white
NOUN 90%

Cluster 41
can
could
might
would
it
there
MD 66.67%

Cluster 42
cent
NOUN 100%

Cluster 43
century
industry
market
NOUN 100%

Cluster 44
certain
further
good
little
very
human
real
ADJ 71.43%

Cluster 45
change
increase
NOUN 100%

Cluster 46
child
woman
figure
line
position
form
use
problems
results
NOUN 100%

Cluster 47
clear
true
close
hard
ADJ 100%

Cluster 48
come
go
help
turn
PRES 100%

Cluster 49
committee
council
government
party
development
education
food
water
law
music
NOUN 100%

Cluster 50
concerned
particularly
shown
PAST 66.67%

Cluster 51
control
experience
policy
later
times
today
cut
set
play
NOUN 88.89%

Cluster 52
day
night
days
years
light
paper
table
time
year
NOUN 100%

Cluster 53
de
john
whose
NOUN 33.33%

Cluster 54
did
does
may
must
will
MD 40%

Cluster 55
different
high
large
long
new
old
young
full
short
ADJ 88.89%

Cluster 56
difficult
necessary
possible
ADJ 100%

Cluster 57
do
know
say
tell
PRES 100%

Cluster 58
doing
found
taken
done
seen
PAST 80%

Cluster 59
door
room
hand
office
side
NOUN 100%

Cluster 60
doubt
fact
NOUN 100%

Cluster 61
dr
miss
mr
sir
mrs
NOUN 100%

Cluster 62
each
those
several
DET 66.67%

Cluster 63
early
east
south
following
united
ADJ 60%

Cluster 64
either
except
forward
given
made
used
ADJ 66.67%

Cluster 65
end
top
NOUN 100%

Cluster 66
england
god
NOUN 100%

Cluster 67
english
lord
president
hall
road
street
NOUN 83.33%

Cluster 68
enough
nothing
something
indeed
PRON 50%

Cluster 69
example
NOUN 100%

Cluster 70
far
much
so
well
soon
mean
ADV 83.33%

Cluster 71
father
mother
wife
NOUN 100%

Cluster 72
feet
hands
own
NOUN 66.67%

Cluster 73
felt
saw
PAST 100%

Cluster 74
few
great
small
special
moment
ADJ 60%

Cluster 75
find
see
show
look
talk
PRES 100%

Cluster 76
first
last
present
next
second
five
three
ADV 42.86%

Cluster 77
gave
took
PAST 100%

Cluster 78
give
keep
leave
let
pay
PRES 100%

Cluster 79
group
system
value
NOUN 100%

Cluster 80
heart
job
story
NOUN 100%

Cluster 81
held
put
kept
turned
PAST 100%

Cluster 82
her
his
my
your
herself
him
them
himself
PRON 100%

Cluster 83
home
place
work
week
least
NOUN 80%

Cluster 84
hope
knowledge
view
NOUN 100%

Cluster 85
hours
months
interest
money
living
working
NOUN 100%

Cluster 86
how
what
why
sure
WH 75%

Cluster 87
idea
sense
NOUN 100%

Cluster 88
important
known
need
reason
ADJ 50%

Cluster 89
kind
type
NOUN 100%

Cluster 90
knew
says
PAST 50%

Cluster 91
level
rate
matter
result
NOUN 100%

Cluster 92
ll
shall
should
MD 100%

Cluster 93
looked
stood
looking
PAST 66.67%

Cluster 94
main
whole
same
ADJ 66.67%

Cluster 95
means
members
part
terms
NOUN 100%

Cluster 96
n't
never
quite
really
ADV 75%

Cluster 97
number
NOUN 100%

Cluster 98
particular
various
ADJ 100%

Cluster 99
seem
seemed
seems
PRES 66.67%

Cluster 100
thing
NOUN 100%