A METHOD FOR IMPROVING AUTOMATIC WORD CATEGORIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMIN ERKAN KORKMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

SEPTEMBER 1997

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Tayfur Öztürk
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Fatoş Yarman Vural
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Ass. Prof. Dr. Göktürk Üçoluk
Supervisor

Examining Committee Members

Prof. Dr. İbrahim Akman

Assist. Prof. Dr. Göktürk Üçoluk

Assist. Prof. Dr. Cem Bozşahin

Assist. Prof. Dr. İbrahim Hakki Toroslu

Assist. Prof. Dr. İlyas Çiçekli

ABSTRACT

A METHOD FOR IMPROVING AUTOMATIC WORD CATEGORIZATION

Korkmaz, Emin Erkan
MS., Department of Computer Engineering
Supervisor: Ass. Prof. Dr. Göktürk Üçoluk

September 1997, 57 pages

In this thesis study a new approach to automatic word categorization which improves both the efficiency of the algorithm and the quality of the formed clusters is presented. The unigram and the bigram statistics of a corpus of about two million words are used with an efficient distance function to measure the similarities of words, and a greedy algorithm to put the words into clusters. The notions of fuzzy clustering like cluster prototypes, degree of membership are used to form up the clusters. Different distance metrics are analyzed using the algorithm. Empirical comparisons are made in order to support the discussions proposed for the type of distance metric that would be most suitable for measuring the similarity between linguistic elements. The algorithm is of unsupervised type and the number of clusters are determined at run-time.

Keywords: Word Categorization, Fuzzy Logic, Distance Metric

ÖZ

OTOMATİK KELİME SINIFLANDIRMA ALGORİTMASINI GELİŞTİRMEK İCİN YENİ BİR YÖNTEM

Korkmaz, Emin Erkan
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Ass. Prof. Dr. Göktürk Üçoluk

Eylül 1997, 57 sayfa

Bu tez çalışmasında *Otomatik Kelime Sınıflandırma* algoritmasını geliştirmeye yönelik bir çalışma yürütülmüştür. Çalışma hem oluşan sınıfların kalitesinin arttırılması hem de algoritmanın hızlandırılmasına yönelik olmuştur. Yaklaşık iki milyon kelimenin bulunduğu İngilizce metinlerden tekli ve çiftli kelime frekansları çeşitli benzerlik fonksiyonları kullanılarak sınıflandırma işlemine tabi tutulmuştur. Küme çekirdekleri, üyelik derecesi gibi bulanık mantığa ait yöntem ve kavramlar bu çalışmada yer almıştır. Doğal dil elemanları arasındaki uzaklığı ölçebilecek en uygun uzaklık fonksiyonunun seçimi üzerine fikirler öne sürülmüştür. Yürütülen tartışmalar algoritma üzerinde denenen değişik uzaklık fonksiyonlarıyla elde edilen sonuçlara göre yapılandırılmıştır. Kullanılan algoritma dışarıdan herhangi bir bilgi almadığı gibi oluşan sınıf sayısı algoritma çalışırken ortaya çıkmaktadır.

Anahtar Kelimeler: Sınıflama, Bulanık Mantık, Uzaklık Fonksiyonu

ACKNOWLEDGMENTS

First of all, I would like to thank Göktürk Üçoluk for his valuable contributions, corrections and discussions.

Also, I would like to thank Cem Bozşahin, the head of the Laboratory for the Computational Studies of Language (LcsL), and other members of the laboratory for their support and for providing the hardware and software resources of this laboratory for my research.

Lastly, I would like to thank all of my friends for their encouragement and support during the preparation of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

I.1 Motivation

How can a human-being acquire natural language? This question denotes an interesting problem in the area of cognitive science. This general problem is related to more than one discipline of science. Linguists are interested in the problem, because they are looking for theories underlying natural language. Philosophers on the other side are examining the relations between language and intelligence. They try to understand the effect of natural language in the process of concept formation and understanding of the world. Language is used for inducing a theory from a set of finite examples. So, philosophers are also interested in natural language as a denotational tool as well as a subject for induction. Furthermore psychologists, working on human-being's behaviors, are looking for relations in the use of language by the human-being and the environment. Lastly computational linguistics are interested in how linguistic knowledge can be acquired by a computer.

Thus it can be claimed that having a theory of acquisition of natural language would aid people to clarify their different applications and theories related to natural language.

Different approaches have been proposed from different disciplines related to language acquisition. However it cannot be claimed that a total theory that can clarify and examine all the processes relevant to natural language acquisition exists. On the other side, researchers working on the subject now underline some aspects of the process.

First of all we know that a child has the capability to map the complex physical signals coming from the outside world onto some representation and by the help of this mapping process, the child can induce the right grammar which lets him/her to understand and produce the utterances. The question at this point is what is the learning mechanism that comes into play? There exist researchers who believe that the concept of "learning" at this point is very closely related to the concept of "behaving in a typical sense". Marcken[1] states that the goal of the learner is to acquire a grammar under which the evidence is 'typical', in a statistical sense. It is a known fact that children can learn language without an explicit, well defined teaching process. By receiving a large number of examples, a child can make up his way through natural language where supervision only takes place while correcting the faulty utterances produced by the child.

Statistical natural language processing is a challenging area in the field of computational natural language learning. Researchers of this field have an approach to language acquisition in which learning is visualized as developing a generative, stochastic model of language and putting this model into practice as stated above [1]. It has been shown practically that the usage of such an approach can yield better performances for acquiring and representing the structure of language.

The availability of large corpora of different natural languages in machine readable form and the availability of computers that can access and process this data have made it possible to test the ideas and theories of statistical natural language researchers empirically. So researchers have been trying to use the regularities of the statistics of natural language to construct a framework for the natural language using this data. Such an approach can be used to find out the structure of language in different levels of abstraction like phonetic or linguistics categories, phrasal structures or even grammatical structures.

Automatic word categorization is an important field of application for this approach where the process is to find out the linguistic categories of words in a natural language. The approach is unsupervised and is carried out by working on n-gram statistics of the language to find out the linguistic categories of words. This process can be visualized as one of the initial steps for the acquisition of natural language in an unsupervised manner. Research in this area points out that it is possible to determine the structure of a natural language by examining the regularities of the statistics of the language [6].

In this thesis a method to improve the automatic word categorization process is presented. The bigram statistics of English corpus, namely the frequencies of word-pair occurences are collected and these statistics are used for the clustering process. An analysis of various distance metrics which serve as the similarity function between words in the clustering process is performed.

A method to improve the classical word categorization algorithms is constructed with two different approaches. First the greedy type algorithm is modified. Also some concepts of fuzzy theory like set prototypes and degree of membership are used for the process. Instead of using *mutual information* which yields a macro-level search among the possible linguistic-categories and which is discussed in detail in sections I.3 and II.1 a bottom up approach which depends on the similarities between single word and word clusters is used. By means of some modifications to the greedy method and fuzzy logic, we tried to improve the method so that the problems of linguistic classification can be overcome.

The reason of choosing English as the test language was due to the morphological structure of this language and the word order sensitivity of English. Since the method is based on frequencies of word-pairs, the linguistic abstraction level used is the word categories. Hence the morphological structure of a word encountered in the corpus is totally ignored. Since the morphological structure of English is not so rich, the lack of a morphological analyzer does not occur a problem regarding our method. However a language with a rich morphological structure like Turkish would need an extra morphological analysis before the clustering process. Also the convergence of the bigram statistics of a language with free word order like Turkish is low compared to a language without free word order.

I.2 Outline of the Thesis

The thesis is based on the idea of finding the structure of natural language in the abstraction level of linguistic categorization. Different approaches have been proposed and various researches have been performed on the subject. The work of different researchers are discussed in the next section I.3. The related search techniques used by the researchers and the basics of their methods are given in order to present the background of the subject.

In Chapter II the classical approach to the process is described in general. The concepts of information theory used for the process are also covered in this chapter. This chapter is followed by the newly proposed method. It covers all the concepts and formulations used to overcome the problems of the classical approach.

In Chapter III the results of the experimentations are given. Discussion of the relevance of the results and conclusion follows in the last chapter.

I.3 Related Work

Automatic word categorization can be visualized as a search problem among the possible linguistic clusters that can be formed using a lexicon space.

Many researchers have been using the concept of *mutual information*, a formulation used in information theory, to search among such possible clusters. The details of this formulation will be discussed in the next chapter.*Mutual information* is just a mathematical formulation to denote the amount of information between two *stochastic variables*.

There exists previous work in which bigram statistics of word pairs are used for automatic word clustering. This approach assumes that taking into consideration bigrams it is possible to define two stochastic variables and thus to determine the amount of mutual information preserved in a natural language corpus. The first stochastic variable will be over the set of words which appear as the first word in the bigrams and the second one over the ones that appear as second. Then the mutual information calculated will denote the amount of information given on the proceeding words in a natural language corpus.

The aim of having such a formulation is to place similarly behaving words in the same clusters. The process can be carried out as follows: When words are placed in clusters there will be a loss in the mutual information. This can be best explained by an example of the extreme case. Consider that all the words in the lexicon space are placed in a single cluster. Then all the bigrams collected from the corpus will produce the information that if you see a word from this 'single' cluster than the proceeding word will be from this 'single' cluster too. Hence no information will be provided by these bigrams. If we calculated the mutual information of such a situation it would come out to be zero denoting 'no information'. If we can succeed in forming wellsuited clusters each corresponding to a linguistic category of the lexicon space then we will have reached the maximum in our mutual information calculation. Hence, for the categorization process, after collecting the bigrams, the cluster organization that will yield the highest mutual information is searched among the possible clusterings. Different search procedures have been used by the researchers to find out such clusters.

There exist researches which conclude that the frequency of single words and the frequencies of occurances of word pairs of a large corpus can give the necessary information to build up the word clusters. Kiss [2] is the first person who has attempted to use bigram statistics for the word categorization process. He used the bigram statistics of the occurances of words in children's stories to classify about 30 words. Finch [3], makes use of bigram statistics for the determination of the weight matrix of a neural network. Brown[5] uses the same bigrams and by means of a greedy algorithm, forms the hierarchical clusterings of words. Finch [3] collects the bigram statistics by a neural network where the units in each layer represent the current word and the ones in the other layer represent the previous or next word. So he interprets a weight-matrix. Here the similarity measure of distribution calculated by the network is the measure of correlation between the bigram statistics of each word and the measure used for this model is the *Spearman Rank Correlation Coefficient* which is found out to be the best empirically.

Genetic algorithms have also been successfully used for the categorization process. Lanchorst [12] uses genetic algorithms to determine the members of predetermined classes. If the *lexicon space* to be clustered is $W = \{w_1, w_2, ..., w_n\}$ consisting of n words, then each *chromosome* in the genetic population would represent the clusters of the words in the lexicon space. That is each *gene* of the chromosome will hold a number denoting the cluster of the word that corresponds to that gene position. The length of the *chromosomes* will be the of the length of the *lexicon space* and the genetic algorithm will search for the chromosome that will yield the optimum categorization. The drawback of his work is that the number of classes is determined previous to run-time and the genetic algorithm only searches for the membership of those classes.

McMahon and Smith [8] also use the mutual information of a corpus to find the hierarchical clusters. However instead of using a greedy algorithm they use a top-down approach to form the clusters. Firstly by using the mutual information the system divides the initial set containing all the words to be clustered into two parts and then the process continues on these new clusters iteratively. Also they use structural tags to keep track of the cluster-hierarchy knowledge of each word. In their model each word is represented as an $s - bit$ number where each bit holds the information for various levels of classification. So given a word with its structural tag all $s-level$ classifications of this word can be accessed immediately. Similar statistical methods to derive word categories have also been used by Brill et al.[4], Kneser & Ney[10] and Hughes[9].

Statistical NLP methods have been used also together with other methods of NLP. Wilms [16] uses corpus based techniques together with knowledge-based techniques in order to induce a lexical sublanguage grammar. Machine Translation is another area where knowledge bases and statistics are integrated. Knight[11] aims to scale-up grammar-based, knowledge-based MT techniques by means of statistical methods. They argue that knowledge based machine translation *(KBMT)* systems can only yield high quality in narrow domains and statistical natural language processing systems can fill the gap for the problems of *KBMT*, like acquiring knowledge resources.

CHAPTER II

WORD CATEGORIZATION

The words of natural language can be visualized as belonging to two different sets. Namely the closed class and the open class. New open class words can be added to the language as the language progresses, however the closed class is fixed and no new words are added to it. For instance, prepositions belong to the closed class. However, nouns are in the open class, since new nouns can be added to the language. The most frequently used words of natural language, usually, belong to the closed class.

Zipf[17] was one of the early researchers in statistical language models. His work states that only 2% of the words of large English corpus is used to form the 66% of the total corpus. Therefore, it can be claimed that by working on a small set which consists of frequent words, it is possible to build a framework for the whole natural language.

N-gram models of language are commonly used to build up such a framework. An N-gram model can be formed by collecting the probabilities of word streams $\langle w_i | i = 1 \ldots n \rangle$ where w_i is followed by w_{i+1} [14]. These probabilities will be used to form the model where we can predict the behavior of the language up to n words.

There exist current research that uses bigram statistics for word categorization in which probabilities of word pairs of the text are collected and processed. These bigram probabilities provide us the information that a language preserves in the word couples. Assume that the bigram statistics of the following two sets of word couples are collected from a natural language corpus. Let the first set be $\mathcal{S}_1 = \{(he, were), (he, are), (she, were), (she, are)\}$ and the second one be $\mathcal{S}_2 = \{(he, was), (he, is), (she, was), (she, is)\}$. It is obvious that the frequencies of the bigrams in the first set will be close to zero since it is not much possible to encounter the words *were* or *are* after the words *he* or *she*, however the frequencies for the second one will be larger than zero since it is likely to encounter such word couples in English. Informally, this information would denote that when we encounter *he* or *she* in a sentence then the next word will probably be from the set $\{was, is\}$, where it is not much possible to have a successor word from the set $\{were, are\}$. Making use of this idea, if we collect the frequencies of word couples from a large corpus, then it would be possible the construct a framework for the language.

II.1 Mutual Information

As stated in the related work part these n-gram models can be used together with the concept of mutual information to form the clusters. Mutual information is the mathematical formulation for the information between two stochastic variables. If the stochastic variables are totally independent from each other then this formulation will result in a zero. On the other hand, if there is an information passed from the first one to the second one, then this will reflect as an increase in the mutual information. *Mutual Information* is based on the concept of *entropy* which can be defined informally as the unpredictability of a stochastic experiment. Let X be a stochastic variable defined over the set $X = \{x_1, x_2, ..., x_n\}$ where the probabilities $P_X(x_i)$ are defined for $1 \leq i \leq n$ as $P_X(x_i) = P(X = x_i)$. The entropy of X which is named as, $H(X)$ is defined by:

$$H(X) = -\sum_{1 \leq i \leq n} P_X(x_i) \ln P_X(x_i) \tag{II.1}$$

The unit of this formulation depends on the base of the logarithm used. If the base is 2 than the *entropy* is said to be in *[bits]*; if the base is e, then it is in *[nats]*. The base of logarithms used in the calculations in this thesis is e, and will be denoted by ln.

If Y is another stochastic variable than the *mutual information* between these two stochastic variables is defined as:

$$I(X : Y) = H(X) + H(Y) - H(X, Y) \tag{II.2}$$

Here $H(X, Y)$ is the joint entropy of the stochastic variables X and Y. The *joint entropy* is defined as:

$$H(X,Y) = - \sum_{1 \le i \le n} \sum_{1 \le j \le m} P_{xy}(x_i, y_j) \ln P_{xy}(x_i, y_j) \tag{II.3}$$

Where $P_{xy}(x_i, y_j)$, the *joint probability*, is defined as

$$P_{xy}(x_i, y_j) = P(X = x_i, Y = y_j) \tag{II.4}$$

Obviously, this is the probability of having at the same moment $X = x_i$ and $Y = y_i$.

Given a lexicon space $W = \{w_1, w_2, ..., w_n\}$ consisting of n words to be clustered, we can use the concept of mutual information for the 'bigram' statistics of a natural language corpus. In this formulation X and Y are defined over the sets of the words appearing in the first and second positions of the bigrams respectively. Assume that the frequencies of the word couples are placed in a two dimensional matrix N. Then each element of the matrix, N_{ij} would hold the number of times the ordered word pair $\langle w_i, w_j \rangle$ which occurs in the corpus. So by using the data in the matrix we can easily compute the number of appearance for the word w_i ($N_{i*} = \sum_{1 \le j \le n} N_{ij}$), or number of appearance for the word w_j ($N_{*j} = \sum_{1 \le i \le n} N_{ij}$), or the total number of word pairs ($N_{**} = \sum_{1 \le i \le n} \sum_{1 \le j \le n} N_{ij}$). So we have the following probability formulations:

$$P(\mathbf{X} = w_i) = p_x(w_i) = N_{i*}/N_{**} \tag{II.5}$$

$$P(\mathbf{Y} = w_j) = p_y(w_j) = N_{*j}/N_{**} \tag{II.6}$$

$$P(\mathbf{X} = w_i, \mathbf{Y} = w_j) = p_{xy}(w_i, w_j) = N_{ij}/N_{**} \tag{II.7}$$

Having the above formulations, it is now possible to reformulate the *mutual information* for the bigram statistics of natural language as follows:

$$I(X:Y) = \sum_{1 \le i \le n} \sum_{1 \le j \le n} p_{xy}(w_i, w_j) \ln \frac{p_{xy}(w_i, w_j)}{p_x(w_i) p_y(w_j)} \tag{II.8}$$

After making the necessary calculations our formula turns out to be:

$$I(X:Y) = \sum_{1 \le i \le n} \sum_{1 \le j \le n} \frac{N_{ij}}{N_{**}} \ln \frac{N_{ij} N_{**}}{N_{i*} N_{*j}} \tag{II.9}$$

This formulation denotes the amount of linguistic knowledge preserved in the bigram statistics. Having this formulation in hand, the process for linguistic categorization becomes a search operation for the cluster organization that will yield the highest mutual information among the possible clusterings. As stated in section I.3 different search procedures can be used to find out the optimal clustering. However, this approach has some drawbacks which prevent the algorithm to form high quality clusters. In the following section II.2 these drawbacks and our proposed method to increase the quality of the formed clusters are presented in detail.

II.2 Clustering Approach

When the mutual information is used for clustering, the process is carried out somewhat at a macro-level. Usually search techniques and tools are used together with the mutual information in order to form some combinations of different sets each of which is then subject to some *validity test*. The idea used for the validity testing process is as follows. Since the mutual information denotes the amount of probabilistic knowledge that a word provides on the succeeding word, if similar behaving words would be collected in the same cluster than the loss of mutual information would be minimal. So the search is among possible alternatives for sets or clusters with the aim to obtain a minimal loss in mutual information.

Though this top-to-bottom method seems theoretically possible, it is timewise not feasible, hence, in the presented work a different approach, which is bottom-up is used. In this incremental approach, set prototypes are built and then combined with other sets or single words to form larger ones. The method is based on the similarities or differences between single words rather than the mutual information of a whole corpus. In combining words into sets a *fuzzy set* approach is used. It is believed that this serves to determine the behavior of the whole set more properly.

Using this constructive approach, it is possible to visualize the word clustering problem as the problem of clustering points in an n-dimensional space if the lexicon space to be clustered consists of n words. The points which are the words of the corpus are positioned on this n-dimensional space according to their behavior related to other words in the lexicon space. Each $word_i$ is placed on the j^{th} dimension according to its bigram statistic with the word representing the dimension, namely w_j. So the degree of *similarity* between two words can be defined as having close bigram statistics in the corpus. Words are distributed in the n-dimensional space according to those bigram statistics. The idea is quite simple: Let w_1 and w_2 be two words from the corpus. Let Z

be the stochastic variable ranging over the words to be clustered. Then if $P_{XY}(w_1, Z)$ is close to $P_{XY}(w_2, Z)$ and if $P_{XY}(Z, w_1)$ is close to $P_{XY}(Z, w_2)$ for Z ranging over all the words to be clustered in the corpus, than we can state a *closeness* between the words w_1 and w_2. Here P_{XY} is the probability of occurances of word pairs as stated in section II.1. $P_{XY}(w_1, Z)$ is the probability where w_1 appears as the first element in a word pair and $P_{XY}(Z, w_1)$ is the reverse probability where w_1 is the second element of the word pair. This is the same for w_2 respectively.

In order to start the clustering process, a *distance* function has to be defined between the elements in the space. The distance function D between two words w_1 and w_2 could be defined as follows:

$$D(w_1, w_2) = D_1(w_1, w_2) + D_2(w_1, w_2) \tag{II.10}$$

where

$$D_1(w_1, w_2) = \sum_{1 \leq i \leq n} \mid P_X(w_1, w_i) - P_X(w_2, w_i) \mid \tag{II.11}$$

and

$$D_2(w_1, w_2) = \sum_{1 \leq i \leq n} \mid P_X(w_i, w_1) - P_X(w_i, w_2) \mid \tag{II.12}$$

D_1 provides the distance due to the two words w_1 and w_2 appearing as the first words in the bigrams and D_2 is the distance when they appear as the second in the bigrams. So the distance function is constructed according to both the preceding and succeeding words of w_1 and w_2 observed in the corpus. Here n denotes the total number of words to be clustered. Since $P_X(w_i, w_j)$ is defined as $\frac{N_{ij}}{N_{**}}$, namely the proportion of the number of occurances of word pair w_i and w_j to the total number of word pairs in the corpus, the distance function for w_1 and w_2 reduces down to:

$$D(w_1, w_2) = \sum_{1 \leq i \leq n} \mid N_{w_1 i} - N_{w_2 i} \mid + \mid N_{i w_1} - N_{i w_2} \mid \tag{II.13}$$

In fact the bigram statistics of each word w_i can be visualized as a vector V_i with components $[v_1, v_2, ...v_n]$. So the $j^t h$ component of V_i will be holding the frequency of $\langle w_i, w_j \rangle$ appearing as word pair in the corpus. The vector V is holding the bigram statistics of word w_i with its succeeding words. A similar vector U_i on the preceding words of w_i can be defined in which the j^th component is holding the similar frequency of $\langle w_j, w_i \rangle$. With the definition of these vectors the distance between two words can be formulated as the distance between them. The distance metrics that can be used between two vectors are discussed in the following section II.2.1.

II.2.1 Distance Metrics

Different distance metrics have been proposed by mathematicians that can be used to formulate the similarity between vectors. Five of them are examined and used for this study. The first one is the *Manhattan Metric* which just calculates the absolute difference between the elements of two vectors. It is defined as:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{1 \leq i \leq n} \mid x_i - y_i \mid \tag{II.14}$$

Here $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ and $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ are the two vectors defined over $\mathcal{R}^n$. This distance metric is the one described as the distance function in the previous section II.2.

Another metric defined is the *Euclidean Metric*:

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{1 \leq i \leq n} (x_i - y_i)^2} \tag{II.15}$$

The formulation of the angle between two vectors is also used in this thesis as a distance metric. If θ is the angle between the two vectors $\mathbf{x}$ and $\mathbf{y}$,then $\cos\theta$ is calculated by:

$$\cos\theta = \frac{\mathbf{a}' \cdot \mathbf{b}}{\mid \mathbf{a} \mid\mid \mathbf{b} \mid} = \frac{\sum_{1 \leq i \leq n} a_i b_i}{[\sum_{1 \leq i \leq n} {a_i}^2]^{\frac{1}{2}} [\sum_{1 \leq i \leq n} {b_i}^2]^{\frac{1}{2}}} \tag{II.16}$$

Since the components of the vectors of our case are corresponding to the frequencies of words, they will be non-negative. So the angle between the two vectors will be between $0°$ and $90°$. Since $\cos 0°$ is unity and $\cos 90°$ is zero, a distance metric between the two vectors can be defined as:

$$D(\mathbf{x}, \mathbf{y}) = 1 - \cos\theta \tag{II.17}$$

This distance metric will give us a number from the closed interval $[0, 1]$. Zero denoting that the two vectors are overlapping and one denoting that there is an angle of 90° which is the highest difference between the vectors.

The fourth distance metric used for the similarity function is the *Spearman Rank Correlation Coefficient*[6] . This metric is based on the difference between the *ranks* of two vectors rather than the difference between their elements. The metric is defined as:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{1 \leq i \leq n} ({R_i}^x - {R_i}^y)^2 \tag{II.18}$$

Here $\mathbf{x}$ and $\mathbf{y}$ are again two vectors as defined above. ${R_i}^x$ nd ${R_i}^y$ are the ranks of the corresponding two vectors. The rank is calculated for our case by normalizing the vectors in the interval $[0, 1]$. The component with the highest value among the components of the vector takes the value 1. If there are n components in the vector, the one with the second highest value will correspond to the number $1 - (1/n)$ and so on. The smallest value will correspond to zero.

The first two of the above metrics depend on the absolute difference between the values of the vector components. However it is a known fact that though some words belong to the same linguistic category, frequencies of them may be totally different. Some concepts are frequently encountered in real life and some not. This results in a difference between the frequencies of various words in language. For instance the word *go* has a very high frequency compared to many other verbs like *appreciate, stalk* and so on, but still we have to cluster *go* with those low frequency verbs. If we use a distance metric based only on the absolute differences of vectors, like the *Euclidean Metric* or *Manhattan Metric*, the distance calculated between high frequency and low frequency words would be high, which is undesired. Therefore when comparing a high frequency word with a low frequency one, we should be able to determine if the difference is caused by some regular magnitude difference. A similarity can exist between the corresponding values when this magnitude difference is discarded. An other example to such a situation could be the similarity between *good* and *preferred.* These two words are from the same linguistic category and they are used frequently in the same context in sentences. However the frequency of *good* is very high compared to *preferred.* Without having a distance function to compensate for, it is not possible to overcome the errors introduced by having different frequencies for words from the same linguistic category. This acts as a considerable factor disturbing the quality of formed clusters.

Having this in mind, the metrics *Spearman Rank Correlation Coefficient* and the *Angle Metric* are used as the distance function. These two seem to discard the magnitude difference between the components of two vectors while calculating the similarity. Such a comparison seems to be more suitable for evaluating the similarity of linguistic elements. In the *Spearman Rank Correlation Coefficient* the vectors are normalized into the closed interval $[0, 1]$. So the vectors are similar if the change from one component to the next is similar, regardless of the differences in the values. We have a similar comparison for the *Angle Metric*. When this metric is used, the length of the two vectors compared may be totally different which reflects a fact of having totally different frequencies in natural language corpora. Regardless of the magnitude of the vectors, if the angle between the two vectors is small, they will be considered similar.

However the improvement obtained using these two metrics has not been so significant. It is believed that the two approaches, that is, comparing the bigram statistics absolutely and making somewhat a relative comparison between them are both effective on fetching out different aspects that leads to similarity between linguistic elements. Therefore, using just one of the approaches is not sufficient to increase the quality of formed clusters. Trying to combine these two approaches have emerged the fifth distance metric which is the combination of the *Angle Metric* and the *Euclidean Metric*.

This fifth metric which we call the *Combined Metric* is defined as :

$$D(\mathbf{x}, \mathbf{y}) = (1 - \cos\theta)\left(\sqrt{\sum_{1 \leq i \leq n} (x_i - y_i)^2}\right) \tag{II.19}$$

Here the $\cos\theta$ is cosine of the angle between the two vectors and since in our case the cosine will be a real number in the interval $[0, 1]$, $(1 - \cos\theta)$ will be a value from $[0, 1]$ as explained above. The rest of the formulation gives us the Euclidean distance between the vectors. Multiplying this Euclidean distance with $(1 - \cos\theta)$ has a following effect. If the angle between the vectors is small, meaning that the vectors are relatively similar, the result of the first part of the formulation will be close to zero. This will be a factor decreasing the value coming from the Euclidean distance. So even if the words have different frequencies in the corpora, their relative similarity will be a factor decreasing the absolute distance between them. With this formulation it is believed that both of the factors affecting the similarity between the linguistic categories are taken into account.

An improvement in the quality of the formed clusters has been obtained by using the *Combined Metric*. The usage of these different metrics and their effect on the clustering process are discussed in Chapter III.

II.2.2 Classical Approach: A Greedy Algorithm

Having such a distance function, it is possible to start the clustering process. The first idea that can be used is to form a greedy algorithm to start forming the hierarchy of word clusters. Let the lexicon space to be clustered consist of $\{w_1, w_2, ..., w_n\}$. The greedy algorithm can be expressed as follows:

1) While a single cluster is still not formed containing all the lexicon space.
   - 1.1) While there exist at least one element which does not belong to any set.
     - 1.1.1) Take the first element in the list which does not belong to any set yet.
     - 1.1.2) Form a cluster with this element and its nearest neighbours.
   - 1.2) Determine the statistical behaviour of the newly formed sets.

According to the above algorithm, first w_1 will be taken from the lexicon space and a cluster with this word and its nearest neighbours will be formed. So the lexicon space would be $\{(w_1, w_{s_1}, ..., w_{s_k}), w_i, ..., w_n\}$ where $(w_1, w_{s_1}, ..., w_{s_k})$ is the first cluster formed. The process will be repeated with the first element in the list which does not belong to any set yet, and this is w_i for our case. After the first iteration of the inner loop, all the words will be in a set. To determine the statistical behavior of a newly formed set, the average of the frequencies of its elements is used. In the next iteration these newly formed sets will be considered as single words and they will be clustered forming larger ones. As denoted in the algorithm the outer loop will iterate until a single set is formed that contains all the words in the lexicon space.

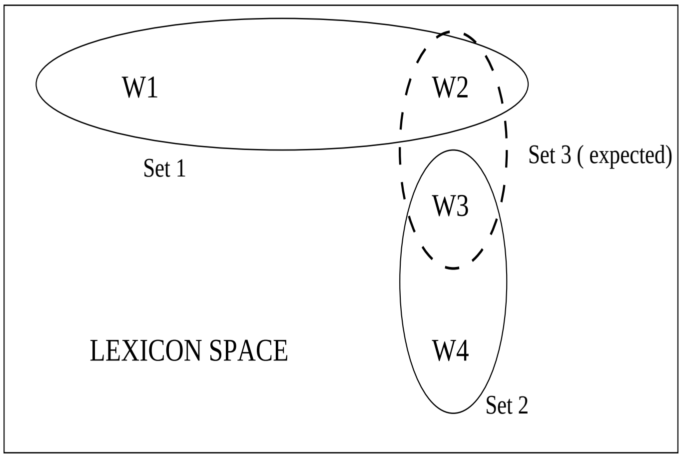
In the early stages of this research such a greedy method was used to form the clusters, however, though some clusters at the low levels of the tree seemed to be correctly formed, as the number of elements in a cluster increased towards the higher levels, the clustering results became unsatisfactory. Also the number of elements in the initial clusters was few. The average number of words in each initial cluster was approximately 2.5. This causes the number of formed sets to increase which makes the cluster hierarchy more complex. Furthermore,to have unsatisfactory number of elements in the initial sets leads to erroneous statistical decisions. This becomes a factor which increases number of the faulty formed clusters towards the upper levels of the cluster hierarchy.

Two main factors were observed as the reasons for the unsatisfactory results.

These were:

- Shortcomings of the greedy type algorithm.
- Inadequacy of the method used to obtain the set behavior from the properties of its elements.

Figure II.1: Example for the clustering problem of greedy algorithm in a lexicon space with four different words.



The greedy method results in a nonoptimal clustering in the initial level. To make this point clearer consider the following example: Let us assume that four words w_1,w_2, w_3 and w_4 are forming the lexicon space. Furthermore let the distances between these words be defined as d_{w_i,w_j}. Then consider the distribution in Figure II.1. If the greedy method first tries to cluster w_1, then it will be clustered with w_2, since the smallest d_{w_1,w_i} value is d_{w_1,w_2}. So the second word w_2 will be captured in the set. The algorithm will continue the clustering process with w_3. At this point, though w_3 is closest to w_2, because it is captured in a set and since w_3 is closer to w_4 than the center of this set is, a new cluster will be formed with members w_3 and w_4. However, as it can be easily seen from Figure II.1 the first optimal cluster to be formed between these four words is the set $\{w_2, w_3\}$.

The second problem causing unsatisfactory clustering occurs after the initial sets are formed. According to the algorithm, the clusters behave exactly like other single words and participate the clustering just as single words do. However to continue the process, the bigram statistics of the clusters should be determined. This means that the distance between the cluster and all the other elements in the search space have to be calculated. One easy way to determine this behavior is to find the average of the statistics of all the elements in a cluster. This method has its drawbacks. If the corpus used for the process is not large, the convergence of the bigram statistics is not high. On the other hand the linguistic role of a word may vary in contexts in different sentences. Many words are used as noun, adjective or falling into some other linguistic category depending on the context. Each word could have a dominant linguistic role and can fall into an initial cluster depending on this role. However if only the average of the bigram statistics of the words are used for determining the statistical behavior of a cluster, the deviations resulting from these different roles prevent determining the dominant statistical role of the cluster. This is denoted as the second drawback of the classical approach.

The clustering process is improved to overcome the above mentioned drawbacks. In the next section II.2.3 the modifications made for the greedy algorithm to overcome the first drawback is presented. In section II.2.4 our new method for determining the statistical behavior of a set is presented so that the method can be improved further.

II.2.3 Improving the Greedy Method

The idea to overcome the shortcomings of the greedy type algorithm is to allow words to be members of more than one cluster initially. So after the first pass over the lexicon space, intersecting clusters are formed. For the lexicon space presented in Figure II.1 with four words, the expected third set will also be formed. As the second step these intersecting sets are combined into a single set. Then the closest two words (according to the distance function) in each combined set are found and these two closest words are taken into consideration as the *prototype* for that set. It is assumed that these two *prototype* words form the *centroid* for that set. After finding the *centroids* of all sets, the distances between a member and all the centroids are calculated for all the words in the lexicon space. Since the the notion of *centroid* denotes the two closest words in a cluster for our case, the distance between a member and a centroid is formulated by the average of the distances between the member and the two closest words forming the centroid. If c_1 and c_2 are the two closest words in a set forming the centroid C for that set, then the distance between a member w_i and this set center would be:

$$D(w_i, C) = \frac{D(w_i, c_1) + D(w_i, c_2)}{2} \tag{II.20}$$

Where D is the distance function used for the linguistic elements. To determine the centroid of each cluster we have to find out the distance between all the words in each cluster. So the complexity of this operation will be $\mathcal{O}(N^2)$. Following this, each word is moved to the set where the distance between this member and the set center is minimal. This procedure is necessary since the initial sets are formed by combining the intersecting sets. When these intersecting sets are combined the set center of the resulting set might be far away from some elements and there may be other closer set centers formed by other combinations, so a reorganization of membership is appropriate. Each word will belong to a single cluster finally.

II.2.4 Improving Further: Using Fuzzy Membership

As presented in the previous section the clustering process builds up a cluster hierarchy. In the first step, words are combined to form the initial clusters, then those clusters themselves become members of the process. In order to combine clusters into new ones the statistical behavior of them has to be determined. The statistical behavior of a cluster is related to the bigrams of its members. To determine the dominant statistical role of each cluster the notion of fuzzy membership is used.

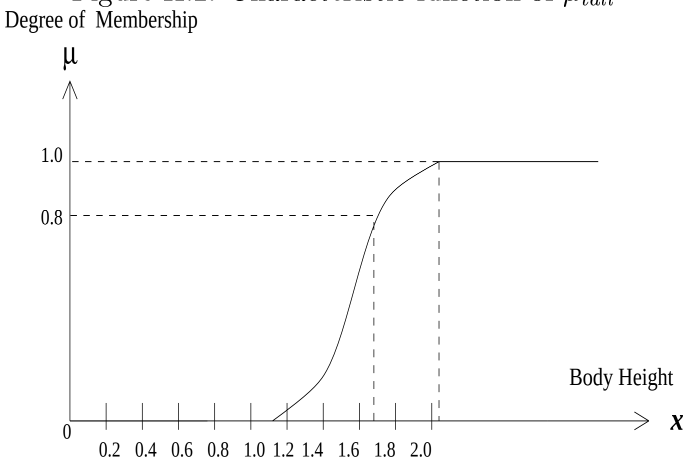
The problem that each word can belong to more than one linguistic category brings up the idea that the sets of word clusters cannot have crisp borderlines and even if a word seems to be in a set due to its dominant linguistic role in the corpus, it can have a degree of membership to the other clusters in the search space. Therefore the concept of fuzzy membership can be used for determining the bigram statistics of a cluster.

The idea of fuzzy membership is used in applications where set membership can not be defined with crisp border. For instance consider the concept of *being tall*. Of course there are some body heights which can be considered as tall like $2.20m$ or being not tall like $1.20m$. However, there also exist body heights which cannot be considered exactly as either ‘tall’, or ‘not tall’. Though there exists a statistical common-sense value, the individual’s decision may vary as well. Hence, instead of using a membership function which takes only one of the values $\{0, 1\}$ (indicating ‘member’:1,‘non-member’:0), we can define a new membership function taking values from $[0, 1]$. So if X denotes the domain of elements of our set being tall, then the membership function (μ) could be defined as:

$$\mu : X \mapsto [0, 1] \tag{II.21}$$

The membership function of the concept ‘being tall’ can then be interpreted as in the figure II.2

Figure II.2: Characteristic function of μ_{tall}



Such a characteristic function can be determined in several ways. One easy way is trying to find out the degree of membership empirically. For our case it is possible to make a small survey to find out the amount of people calling a person with some body height ‘tall’ or ‘not tall’. In our characteristic function the body height $1.70m$ has a degree of membership for being tall as 0.8. this may just denote the percentage of people considering body height $1.70m$ as tall. Obviously it is impossible to apply this approach to all fuzzy concepts. So some mathematical formulations could be used in order to determine the degree of membership of a certain set.

Researchers working on fuzzy clustering present a framework for defining fuzzy membership of elements. Gath and Geva [7] describe such an unsupervised optimal fuzzy clustering. They present the K-means algorithm based on minimization of an objective function. For the purpose of this research only the membership function of the presented algorithm is used. The membership function u_{ij} that is the degree of membership of the i^{th} element to the j^{th} cluster is defined as:

$$u_{ij} = \frac{\mid \frac{1}{d^2(X_i, V_j)} \mid^{\frac{1}{(q-1)}}}{\sum_{k=1}^{K} \mid \frac{1}{d^2(X_i, V_j)} \mid^{\frac{1}{(q-1)}}} \tag{II.22}$$

Here X_i denotes an element in the search space, V_j is the centroid of the j^{th} cluster. K denotes the number of clusters. And $d^2(X_i, V_j)$ is the distance of $X_i th$ element to the centroid V_j of the j^{th} cluster. The parameter q is the weighting exponent for u_{ij} and controls the fuzziness of the resulting cluster. Since we can determine the centroids for the linguistic clusters for our application this formulation is made use in the algorithm to find out the membership degrees of elements with respect to various clusters. These clusters later will correspond to the linguistic categories.

After the degrees of membership of all the elements of all classes in the search space are calculated, the bigram statistics of the classes are derived. To find those statistics the following method is used: For each subject cluster, the bigram statistics of each element is multiplied with its membership value. This forms the amount of statistical knowledge passed from the element to that set. So the elements chosen as set centroids will be the ones that affect a set's statistical behavior mostly. Hence an element away from a centroid will have a lesser statistical contribution.

Let V be a vector of length n holding the statistics of a cluster C where n denotes the number of elements in our lexicon space. V_i will be a frequency denoting the bigram statistics of that cluster with $word_i$. The bigram statistics for each V_i can be formulated by the function S as follows:

$$S(V_i) = \sum_{k=1}^{k=n} u_{kC} \cdot N_{ki} \tag{II.23}$$

Here u_{kC} denotes the membership degree of $word_k$ to the set C and N_{ki} is the frequency of the word couple $\langle w_k, w_i \rangle$. Note that this vector V is formulated for the succeeding words. Similarly the vector for the preceding words can be formulated. These two vectors will then hold the statistical behavior for the cluster C.

CHAPTER III

RESULTS

III.1 The Corpus Used

The algorithm is tested on a corpus formed with online novels collected from the world wide web page of the "Book Stacks Unlimited, Inc." The corpus consists of twelve publicly available on-line novels. These novels include some of the classics *"Tale of two cities; Anna Karina; The invisible man"* and so on. The corpus is passed through a filtering process where the special words and useless characters are filtered out. After this initial process, frequencies of words are collected. The most frequent thousand words are chosen and sent to the clustering process described in the previous sections. The following remarks could be made about the properties of the corpus used.

- The corpus consists of about 1.700.000 words.
- About 38000 different words are encountered in the whole corpus.
- The most frequent word in the corpus is *'the'* with frequency 5%. (The first twenty-five most frequent words in the corpora and their frequencies are presented in Table III.1.)
- The frequency goes down to 0.11% after the most frequent one hundred words, and down to 0.018% after the first five hundred and down to 0.008% after the first one thousand.
- The percentage of the closed class words among the 'most frequent one hundred words' is 87%. This percentage is 60.8% for the most frequent 250 words. This

Table III.1: Frequencies of the most frequent twenty-five words

the	5.002056%
and	3.281249%
to	2.836796%
of	2.561952%
a	2.107116%
in	1.591189%
he	1.533916%
was	1.419838%
that	1.306431%
his	1.124362%
it	1.061797%
her	1.056439%
had	0.992802%
with	0.928294%
you	0.913490%
as	0.830963%
for	0.819241%
she	0.815422%
not	0.742675%
but	0.709919%
at	0.703087%
said	0.637373%
on	0.629067%
him	0.612387%
be	0.595038%
is	0.507286%
have	0.493018%
my	0.456310%

percentage goes downto 2.4% for the least frequent 250 words among the most frequent thousand ones.

- The most frequent thousand words form the 70.4% of the whole corpus
- The percentage goes up to 77% if the next most frequent thousand is added to the lexicon space.
- Examples of the least frequent words in the corpus which show up only once (0.000067%) are *"incumbrances, irretrievably, wormwood, tobaccostopper and coarser"*

The clustering process builds up a tree hierarchy having words at the leaves and clusters at the inner nodes. The root node denotes the largest class containing all the lexicon space. At each level a different cluster organization could be observed for the lexicon space. The leaves are the clusters of single words formed by the first pass of the algorithm. At the upper levels each node represent clusters formed as various combinations of clusters at the lower levels.

The test of the algorithm consists of five parts. In each part a different distance metric is used. These metrics were *Manhattan Metric*, *Euclidean Metric*, *Spearman*

Rank Correlation Coefficient and the *Angle Metric* which is based on the angle between the vectors as explained in section II.2.1. The *Combined Metric*, a combination of the two metrics *Euclidean Metric* and the *Angle Metric* is also used for the testing process as the fifth one.

As was explained in section II.2.1, among these four distance metrics the first two, the *Manhattan Metric* and the *Euclidean Metric* are based on the absolute difference between the components of the vectors. The *Angle Metric* and the *Spearman Rank Correlation Coefficient* tries to disregard the magnitudial difference between the components of the two vectors and seem to make a relative comparison between them. Lastly, the newly proposed *Combined Metric* tries to combine these two aspects.

In the next section the results obtained with the *Manhattan Metric* will be discussed and then comparisons will be made between these results and the ones obtained with the other metrics.

III.2 Empirical Comparisons

The *Manhattan Metric* is based on the absolute difference between the component values of the two vectors. So it might be considered as the poorest metric used for the results. The outcomes of this test will serve as a lower bound for our algorithm that modifies the greedy algorithm and uses the fuzzy concepts.

Some linguistic categories inferred by the algorithm formed using the Manhattan Metric are listed below:

- prepositions(1): by with in to and of
- prepositions(2): from on at for
- prepositions(3): must might will should could would may
- determiners(1) : your its our these some this my her all any no
- prepositions(4): between among against through under upon over about
- adjectives(1) : large young small good long
- nouns(1) : spirit body son head power age character death sense part case state
- verbs(1) : exclaimed answered cried **says** knew felt said **or is was** saw did asked gave took made thought **either** told **whether** replied **because though how** repeated open remained lived died lay **does why**

- verbs(2) : shouted wrote showed spoke **makes** dropped struck laid kept held raised led carried sent brought rose drove threw drew shook talked **yourself** listened wished meant **ought seem seems** seemed tried wanted began used continued returned appeared **comes knows** liked loved
- adjectives(2) : sad wonderful special fresh serious particular painful terrible pleasant happy easy hard sweet
- nouns(2) : boys girls gentlemen ladies
- adverbs(1) : scarcely hardly neither probably
- verbs(3) : consider remember forget suppose believe say do think know feel understand
- verbs(4) : keeping carrying putting turning **shut** holding getting hearing knowing finding drawing leaving giving taking making having being seeing doing
- nouns(3) : streets village window evening morning night middle rest end road sun garden table room ground door church world name people city year day time house country way place fact river next earth
- nouns(4) : beauty confidence pleasure interest fortune happiness **tears**

The ill-placed members in the clusters are shown above using bold font. The clusters above represent the linguistic categories with a high success rate ($\approx$ 91%). This is approximately the same for the other categories which are not presented here except one. The main problem of the cluster organization obtained using this metric was a single large, faulty cluster which contained different linguistic categories. This was a factor decreasing the success rate of the overall clustering process. However if we discount this faulty large cluster we could evaluate the success rate to be about 90% for the initial clusters. When the classical algorithm using mutual information and a greedy method has been used on the same lexicon, the average number of elements in the initial clusters was approximately 2.5 which is 13.4 for the clusters presented above. Therefore the above clusters denote a high progress in the number of words correctly combined in the initial clusters. When higher number of words are collected in these initial ones, the complexity of the hierarchy obtained decreases. This is a factor increasing the success of the clustering process.

Also some semantic relations in the clusters can be observed. Group nouns(2) is a good example for such a semantic relation among the words in a cluster. Obviously if the words are semantically related, then they will be behaving similarly in sentences. So, probably will have very close bigrams. This is usually closer than the words with no semantic relation. Therefore semantically related words are usually observed in these linguistic categories.

Table III.2: Comparison of cluster hierarchies obtained with different metrics.

Test Criteria	**Manhattan Metric**	**Angle Metric**	**Euclidean Metric**	**Spearman Rank Correlation Coefficient**	**Combined Metric**
# of initial clusters	60	169	132	185	171
# of elm. in the initial clusters	16.6	5.9	7.56	5.4	5.8
Depth of the tree	8	3	9	11	11
Location of leaves	$5^{th} and$ $6^{th} levels$	$3^{th} level$	$7^{th} and$ $8^{th} levels$	$9^{th} and$ $10^{th} levels$	$9^{th} and$ $10^{th} levels$
# of nodes on the second level	18	39	35	41	37

The same clustering process is repeated using the distance metrics presented in section II.2.1 on the same lexicon space. The above clusters are the initial categories obtained in the hierarchy. When other metrics are used for the distance function, the large faulty cluster obtained with the first *Manhattan Metric* disappeared. The success rate obtained for the initial clusters is again about 90%, when these new distance metrics are used. However the main difference between the different metrics used was in the structure of the cluster hierarchy obtained. Different metrics lead to different ways of combinations of initial sets into larger ones. This means a different hierarchy is being formed.

The details of the different cluster hierarchies obtained are given in table III.2. When the properties presented in this table are examined, the hierarchy formed by the *Manhattan Metric* seems to be the simplest one having the minimum number of initial clusters. This is due to the large faulty cluster formed with this metric. This faulty cluster is holding about 300 words. Although it is the nouns which are mainly located in this cluster, verbs, adjectives and some prepositions are also added to it. Though the other clusters seem to be well formed it cannot be claimed that the results obtained are satisfactory with such a large faulty cluster to hand. It is belived that the cause for it is that the metric is not powerful enough to make clear distinctions. According to the algorithm the intersecting clusters are formed and they are combined into one cluster. If the distance metric used is not powerful enough, although we may get a very high success rate for some of clusters, certain clusters with different linguistic roles will seem to be overlapping and they might be combined into one cluster in the first pass of the algorithm. This problem disappeared when the other distance metrics were used. The algorithm has been able to divide this large faulty cluster into smaller ones by using the new metrics. This is denoted as the reason for the increase in the number of initial clusters when the new ones are used. The properties of the hierarchies presented in table III.2 seem to be similar to each other. Only the depth of the tree formed with the *Angle Metric* differentiates with the other ones. This is because more initial clusters are combined into one on the second level in the hierarchy obtained with this metric. This brings in an increase in the number of ill-structured clusters on the second level overcombining distinct linguistic categories.

In Table III.2 two parts of the cluster hierarchy obtained using the *Manhattan Metric* are presented. The first one collects a set of nouns from the lexicon space. However the second one is somewhat ill-structured namely two prepositions, two adjectives and a verb cluster are combined into one. Such problems are common for this metric's cluster organization in different regions of the hierarchy. A perfect classification has not been observed after the first level. It could be denoted that solving this problem presented here is the main progress obtained when the combined metric is used. In Table III.4 and Table III.2 two parts of the cluster hierarcy obtained using the *Combined Metric* are presented. In table III.4 94 nouns coming from different initial clusters are combined into the same part of the cluster hierarchy. Ony one cluster seems to be misplaced in this part. This is an adjective cluster. Also we have some some incorrect ones in the initial clusters. However it is believed that this is an important improvement compared to earlier results. Also in the second table III.2 67 different verbs are collected. They are all present tense verbs and no single misplaced word exists in this part of the hierarchy. This is another well-formed part of the cluster organization. Such a progress was not observed when any of the other metrics were used.

Table III.6 exhibits the improvement obtained using the *Combined Metric*. Maximum number of words correctly classified for some linguistic categories are shown in this table. Obviously there are other clusters having elements from the same linguistic categories in different parts of the hierarchy. This table makes a comparison of the maximum numbers in order to analyze the process of combination of initial clusters. Gathering *nouns* and *auxilaries* seem to be carried out better using the *Manhattan Metric*, however the number of initial clusters used to form these clusters is larger when the *Combined Metric* is used. There is a big difference between the two. For instance 12 *present perfect verb* classes are combined successfully when the *Combined Metric* is used, but only 3 of them were combined for the other metric. For *adjectives* this is

Table III.3: Examples from the cluster hierarchy for the Manhattan Metric

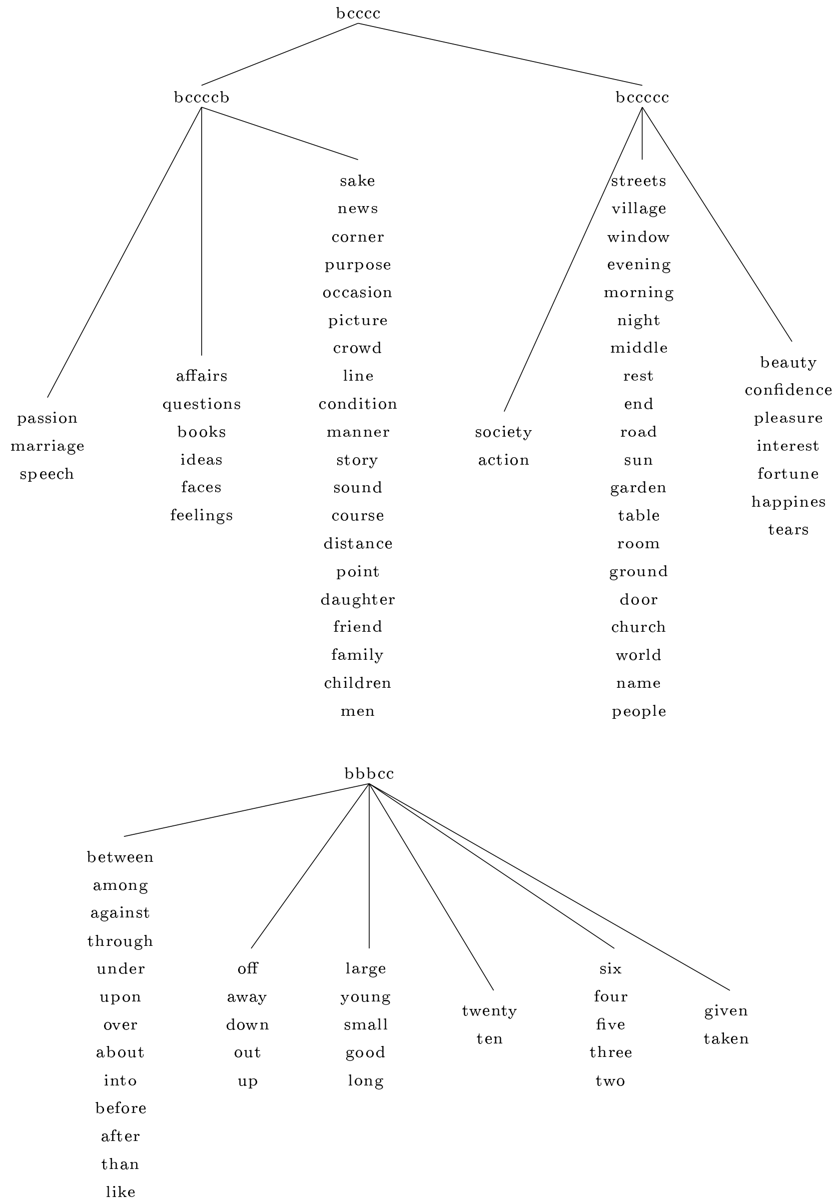
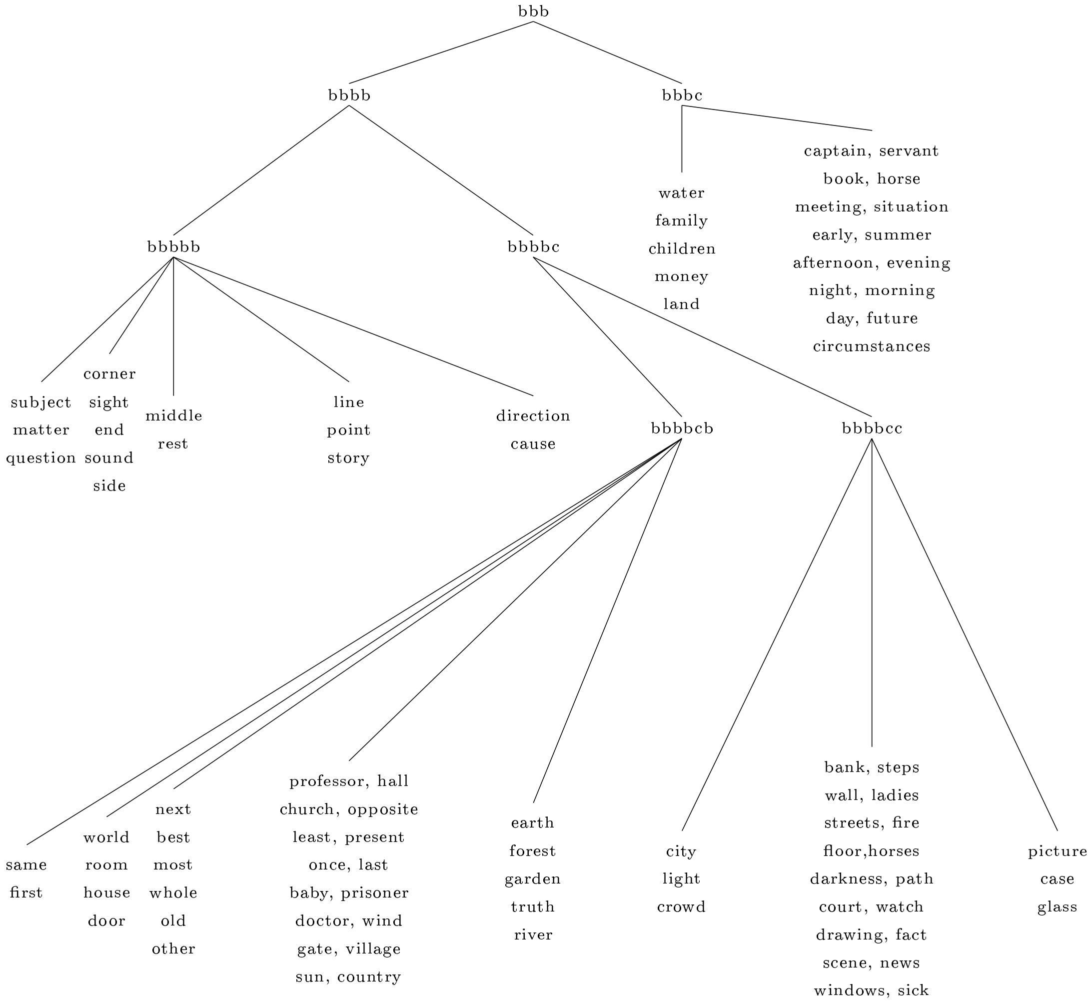


Table III.4: Part of the cluster hiearchy holding nouns



7 to 2, for *past perfect verbs* 5 to 1 and although number of *nouns* collected for the *Manhattan Metric* is larger, number of initial clusters combined using the *Combined Metric* is still larger.

It can be claimed that there is a significant progress in the process of successfully combining the initial clusters formed by the algorithm when the new metric is used. This was the main problem encountered with the *Manhattan Metric* and the other ones.

Table III.5: Part of the cluster hiearchy holding present tense verbs

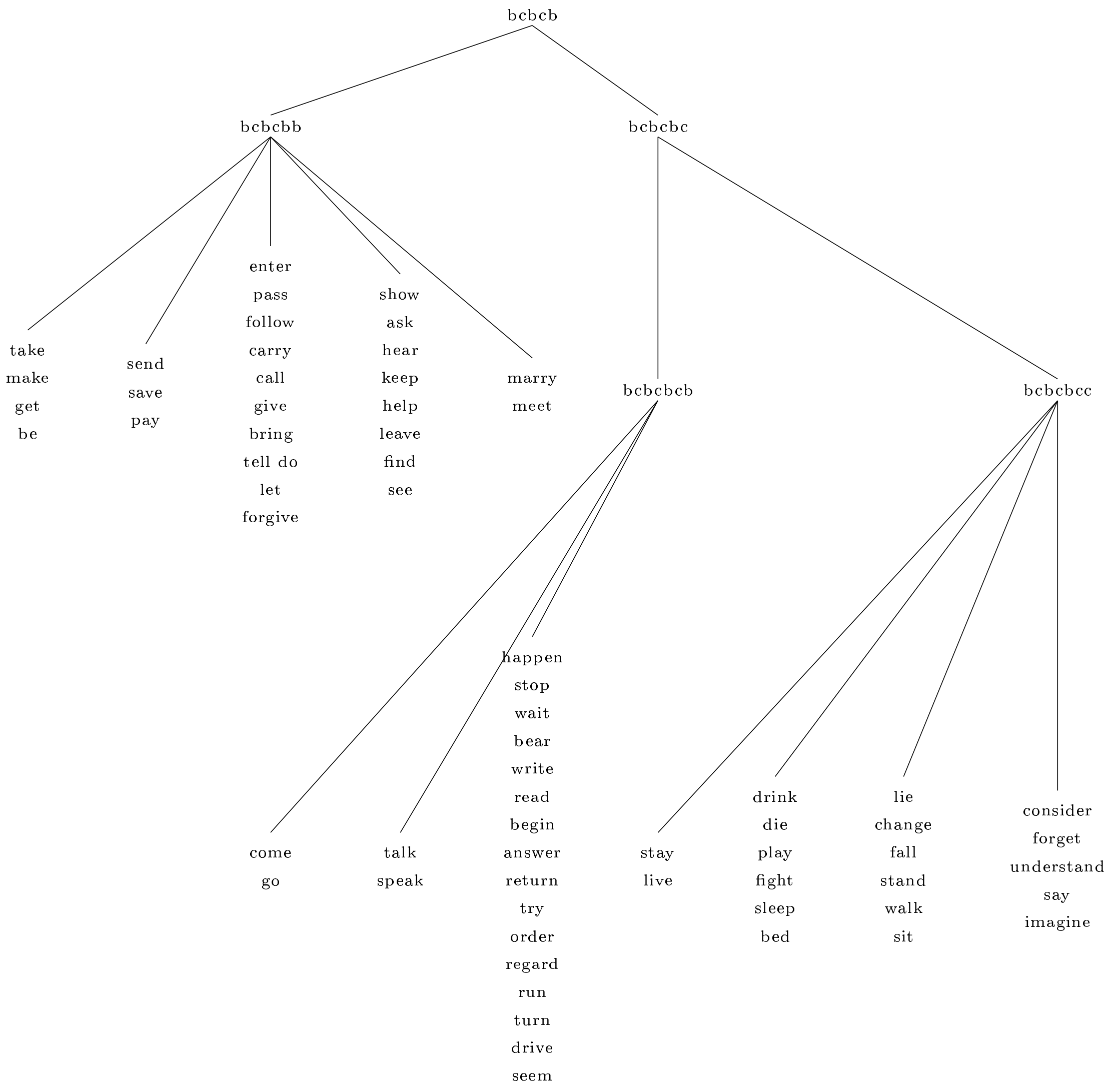


Table III.6: Comparison made between Combined Metric and Euclidean Metric based on the largest number of elements combined in a cluster.

	Combined Metric	**Manhattan Metric**
Nouns		
Largest # of words collected	94	111
Success Rate	91.5%	94.6%
# of initial clusters connected	15	6
Verbs (present perfect)		
Largest # of words collected	67	45
Success Rate	100%	73.3%
# of initial clusters connected	12	3
Verbs (past perfect)		
Largest # of words collected	16	2
Success Rate	100%	100%
# of initial clusters connected	5	1
Adjectives		
Largest # of words collected	68	17
Success Rate	92.6%	100%
# of initial clusters connected	7	2
Adverbs		
Largest # of words collected	9	4
Success Rate	100%	100%
# of initial clusters connected	1	1
Auxiliaries		
Largest # of words collected	7	9
Success Rate	100%	100%
# of initial clusters connected	1	1
Determiners		
Largest # of words collected	16	10
Success Rate	100%	100%
# of initial clusters connected	1	1

CHAPTER IV

DISCUSSION AND CONCLUSION

IV.1 Achievements

This research has focussed on improving automatic word categorization which can be visualized as one of the initial steps for unsupervised language acquisition. It can be claimed that the results obtained in this research are encouraging. The research on such bottom-up statistical approaches has shown that it is possible to apply unsupervised learning algorithms to natural language.

The corpus used for this research has been formed consisting of free on-line novels. In the early stages of the research the classical greedy type algorithm with mutual information has been applied on this corpus. Earlier research has shown that it is possible to use such an algorithm on various corpora. However the same algorithm didn't give a satisfactory result on our corpus. The average number of elements in the initial clusters was only 2.5 and the success in combining these initial clusters into larger ones was low. So new methods are tried to increase the success rate on the same corpus. The algorithm is modified so that instead of using mutual information, a method based on the distance between linguistic elements is developed. The greedy algorithm is modified successfully so that the shortcomings of this method are overcomed. Another inadequacy of the earlier methods was in obtaining the set behavior from the properties of its elements. In the proposed method using fuzzy membership has been a further improvement overcoming this inadequacy.

When this new algorithm has been used, we had an increase in the number of elements in the initial clusters. But the results still seemed to be erroneous. For instance we had a large faulty initial cluster consisting of about 300 words which was undesired. Also the process of combining initial clusters was still a problem for the algorithm. The distance metric initially used *(Manhattan Metric)* was not so powerful. It is believed that the erroneous results like the large faulty cluster were mainly due to the usage of the distance metric. So after this stage, the research has focused on the usage of the distance metric used to measure the similarity between the elements. Different distance metrics proposed have been tested in the algorithm. We were able to develop a special purpose metric *(Combined Metric)* which tries to fetch different aspects that leads to the similarity between linguistic elements. The best result has been obtained using this metric. The algorithm has been able to divide the large faulty cluster into smaller ones and there has been a significant improvement in the process of combining the initial clusters into larger ones. When the *Combined Metric* is used, the success rate in the initial clusters is about 90% and the average number of elements in the initial clusters is about 6. Furthermore, 15 noun, 12 present perfect tense verb, 5 past perfect tense verb and 7 adjective initial clusters have been successfully combined in the same region of the cluster hierarchy which is a significant improvement compared with the earlier results.

IV.2 Critics

The main drawback of the research was the corpus used for collecting the frequencies of words. As explained in chapter III the corpus is formed with on-line novels. So the style of sentences differentiates from author to author. It is believed that this is a factor decreasing the convergence for the information gathered by the bigram statistics of words. Also the size of the corpus ($1.700.000 words$) is another factor causing this decrease. A corpus with crisp borderlines for the grammatical structure of sentences like technical manuals may help to produce better linguistic categories. It was not possible to observe such crisp sentences with literature work. It is believed that with a larger training data which contains crisp sentence structures, an increase in the convergence of frequencies, thus an increase in the quality of clusters can be expected. However, the deviations coming from the implicit structure of natural language would still be existing. To overcome this problem our proposed method is making use of fuzzy membership concept.

Considering the results obtained by the experiments carried out, the following remarks could be made on the linguistic clusters formed. In the initial clusters formed, the success rate obtained is quite satisfactory. However, it was not always possible to to combine these initial clusters into exact linguistic categories. We sometimes encountered different *noun* or *verb* categories at different parts of the cluster hierarchy which actually should be the same. A significant progress has been obtained when the *Combined Metric* has been used for the algorithm. However it was impossible to combine all the initial clusters belonging to the same linguistic category at the same location of the hierarchy. This is mainly due to the the very complex structure of the natural language. The large number of sentence forms in natural language and the fact that many words can be used with different linguistic roles in sentences produces deviations in the information given by the bigrams. Using fuzzy logic is a way to decrease these deviations, however it was not possible to remove them totally.

Although the clusters formed in our study seem to deviate from a perfect linguistic categorization, the results seem to show that natural language preserves the necessary information implicitly for the acquisition of it. A convergence of linguistic categories could be obtained by using the algorithm we have presented. This result is certainly motivating further studies on statistical acquisition of natural language.

The distance function used in the algorithm is the main factor determining the running time of the proposed method. The bigram statistics are obtained from the corpus by scanning through the corpus only once. Therefore, the running time of this process is $\mathcal{O}(N^2)$. The distance function depends on the comparisons between the two vectors' components. The running time for this operation is at worst $\mathcal{O}(N \lg N)$ (due to sorting). Since the distance between each pair of linguistic elements is calculated, the running time of the algorithm rises to $\mathcal{O}(N^2 \lg N)$.

Different distance metrics are used in the algorithm. Two of them are based on the absolute difference between the bigram statistics. For these two metrics, the running time of the algorithm is quite low compared to other ones. Though the complexity of the algorithms were the same, there is an increase in the efficiency due to gain of the time which is consumed by the mathematical operations (like division and multiplication) in the metrics like *Spearman Rank Correlation Coefficient*. However as explained in chapter III using the metrics based on relative differences between the bigram statistics of words improve the quality of clusters since in natural language some words may have very high frequencies and still be in the same clusters with low frequency words. We can get the best results when we combine the two different metrics which weighs individually the two contradicting aspects of similarity, in the same one.

IV.3 Future Work and Possible Application Areas

The implicit information could be used for different applications on natural language. The high success rate obtained in the initial clusters makes it possible to use the algorithm for automatically tagging a natural language text. There are researches who work on tagging natural language text with a probabilistic model [13]. Although the large number of initial clusters is a problem, trying to develop algorithms for automatic acquisition of natural language grammar by means of such a unsupervised algorithm seems to be possible.

Inference of the phrase structure of a natural language could be another application area. Finch [6] again uses the mutual information to find out such structures. Using fuzzy membership degrees could be another way to repeat the same process. To find out the phrases, most frequent sentence segments of some length could be collected from a corpus. In addition to the frequencies and bigrams of words, the statistics for these frequent segments could also be passed to the clustering inference mechanism and the resulting clusters would then be expected to hold such phrases together with the words.

Another important problem that can be encountered working on natural language processing is the *Word Sense Disambiguation*. This is a very hard problem of NLP where different approaches are proposed. Statistical methods are also developed for the problem[15]. This may be visualized as an other application area to apply the algorithm. Modifications and a more developed approach would be needed for such an application. It is very hard to claim that *Word Sense Disambiguation* could be tackled using the information given by the bigram statistics of natural language. Many words are disambiguated using the information passed from some other word which may be located far away from that word. This location may vary depending on the structure of language or the sentence. So probably a larger frame (possibly n-gram) would be needed rather than the bigram statistics in order to capture the necessary information to solve the problem.

Since the corpus used for this research is different then the ones referred to by other studies, it has not been possible to give empirical comparisons between our results and the previous ones. It is also a fact to be mentioned that many researches are not precisely identifying their corpus at all. As further research, testing the algorithm on a well-known corpus like the *Brown Corpus* can be useful for comparing the success of the method with the previous researches.

Computational natural language learning is an area related to different disciplines and has so many application areas directly related to the usage of computers in the daily life. Different approaches could be proposed for the problem. Statistical approach is just one of these which proves natural language preserves a structure that can be tackled by unsupervised, bottom up techniques. However this is certainly not the ultimate solution to the problem at hand. It could be claimed that the technological progress will result in an increase of computer-real world interaction and this will form a base where the natural language applications could easily be developed and tested. It seems that statistical approaches to natural language will be one of the main approaches for such applications in the future. Researches claim that at least for filling the gaps the classical methods have, statistical methods come into play and more powerful systems could be developed using the information that can be gathered by statistical methods.

To conclude, a last remark could be that automatic word categorization is one of the initial steps for the acquisition of the structure of natural language. Similar methods could be used with modifications and improvements to find out more abstract structures in the language and moving this abstraction up to the sentence level successfully might make it possible for a computer to acquire the whole grammar of any natural language automatically.

REFERENCES

[1] Carl G. de Marcken. Unsupervised Language Acquisition PhD Thesis, 1996.

[2] Kiss, G.R. Grammatical Word Classes: A learning process and its Simulation. *Psychology of Learning and Motivation.* **7** 1-41.

[3] S.Finch and N.Charter, Automatic methods for finding linguistic categories. In Igor Alexander and John Taylor, editors, Artificial Neural Networks, volume 2. Elsevier Science Publishers, 1992

[4] Brill, E., D. Magerman, M. Marcus & B. Santorini. Deducing Linguistic Structure from the Statistics of Large Corpora. In DARPA Speech and Natural Language Workshop. Morgan Kaufmann, Hidden Valley, Pennsylvania. 1990

[5] Brown P.F., V.J. Della Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics,* 18(4):467-477, 1992

[6] Finch,Paul Steven. Finding Structure in language. PhD Thesis. Centre for Cognitive Science, University of Edinburgh, 1993.

[7] Gath I, Geva A.B. Unsupervised Optimal Fuzzy Clustering. *IEEE Transactions on pattern analysis and machine intelligence, Vol. 11, No. 7,* July 1989.

[8] John G. McMahon Francis J. Smith. Improving Statistical Language Model Performance with Automatically Generated Word Hierarchies. *Computational Linguistics,* 22(02):217-248, 1996.

[9] Hughes, J. & E.S. Atwell. Automatically Acquiring and Evaluating a Classification of Words. IEE digest 1993/092: Grammatical Inference: Theory, Applications and Alternatives. 1993

[10] Kneser, R. & H. Ney. Forming Word Classes by Statistical Clustering for Statistical Language Modeling. In Proceedings of QUALICO 1, Trier, Germany. 1991.

[11] Knight Kevin, Chander Ishwar, Haines Matthew, Hatzivassiloglou Vasieios, Hovy Eduard, Iida Masayo, Luk Steve, Okumura Akitoshi, Whitney Richard, Yamada Kenji. Integrating Knowledge Bases and Statistics in MT. Proceedings of the 1st AMTA Conference. Columbia, MD. 1994.

[12] M.M. Lankhorst. A Genetic Algorithm for Automatic Word Categorization. In: E. Backer (ed.), Proceedings of Computing Science in the Netherlands CSN'94, SION, 1994, pp. 171-182.

[13] Merialdo B. Tagging English Text with a Probabilistic Model. IEEE International Conference on Acoustics, Speech and Signal Processing, Toronto (Canada), May 1991.

[14] Stanley F. Chen.Building Probabilistic Models for Natural Language. Ph.D. thesis. Technical Report TR-02-96, Center for Research in Computing Technology, Harvard University, 1996.

[15] William Gale, Kenneth Church and David Yarowsky. Work on Statistical Methods for Word Sense Disambiguation. In Proceedings, AAAI Fall Symposium on Probabilistic Approaches to Natural Language, Cambridge, MA, pp. 54-60, 1992.

[16] Wilms, Geert Jan. Automated Induction of a Lexical Sublanguage Grammar Using a Hybrid System of Corpus and Knowledge-Based Techniques. Mississippi State University. PhD Thesis, 1995.

[17] Zipf, G. K. *The psycho-biology of Language.* Boston: Houghton Mifflin. 1935

APPENDIX A

IMPLEMENTATION

The implementation of the algorithm proposed is carried out using *C* programming language in *unix* environment. The basic data structures and the specifications of the main functions used are presented below.

A.1 Data Structures

Some parameters for the algorithm are defined as constants in the implementation. These are:

- #define FREQWORDNUM 1000
- #define WORDNUM 1000
- #define WORDLENGTH 60

Where *FREQWORDNUM* is the number of most frequent words that will form the lexicon space. *WORDNUM* is the number of most frequent words that will be used for collecting the bigram statistics. These two constants are kept the same for the sample runs of the research. However *WORDNUM* could be larger than the *FREQWORDNUM*. It is feasible to gather the statistics of a larger class than the lexicon space to be clustered. *WORDLENGTH* is the maximum number of characters for each word.

The cluster hierarchy formed using the algorithm is represented in the implementation by means of a *embedded list*. Below is the definition of a node belonging to this hierarchy.

- ```
  struct treeelm
    {
     char wordelm[WORDLENGTH];
     double membership;
     int freq;
     int indx;
     struct treeelm *nextpointer;
     struct treeelm *downpointer;
    };
  typedef struct treeelm thetree;

  thetree *treepoint;
  ```

In each node the name for the linguistic element *(wordelm)*, the frequency of it *(freq)*, degree of membership *(membership)* for some cluster and an index *(indx)* used for hashing algorithms are hold together with two pointers. The *downpointer* is the one pointing to the list of the children for this node and *nextpointer* is the one used to construct the list of elements having the same parent with this node.

In order to access an element in the cluster hierarchy a *linear hash* algorithm is used. The following definition is for the array used for the hash algorithm where each element of the array consists of the word itself and a pointer to the position of this word in the cluster hierarchy.

- ```
  struct hasharrayelm
    {
     char wordelm[WORDLENGTH];
     thetree *clusterpointer;
    };
  typedef struct hasharrayelm hasharrelm;
  hasharrelm hasharray[FREQWORDNUM];
  ```

The bigrams of the word couples are kept in the following two dimensional array.

- int matrix[WORDNUM][WORDNUM];

It is possible to access the bigrams of a linguistic element in this matrix using the index kept in the data structure *thetree* denoting a node in the cluster hierarchy.

Another important data structure is the one used for the cluster centroids. According to the algorithm two words represent the center of each cluster, so the following structure is implemented for holding the centroids:

- ```
  struct setcenterelm
    {
     thetree *xpoint;
     thetree *ypoint;
     struct setcenterelm *nextpointer;
    };
  typedef struct setcenterelm setcenter;
  ```

In this implementation *xpoint* and *ypoint* are two pointers showing the place of the two words in the cluster hierarchy. By using the third one, that is *nextpointer*, it is possible to implement a linked list of set centroids.

A.2 Functions and Procedures

A.2.1 Highest Level of Abstraction

The algorithm presented is carried out by the following functions which can be visualized as the ones at the highest level of abstraction.

- void initializeunigram(char corpusname[WORDLENGTH],
  char freqname[WORDLENGTH])
- void readfreqs(char freqname[WORDLENGTH]));
- gatherbigrams(char corpusname[WORDLENGTH])
- void formgroups();
- void printgroups(thetree *mytree,FILE *oflp)

The implementation of the algorithm can be roughly divided into two parts. In the first part, the corpus is scanned through and a file containing the frequencies of
  ```

various words encountered in the corpus is prepared. This is carried out by the first function *initializeunigram* which takes two parameters denoting the name of the file containing the paths for the files forming the corpus and the name of the output file that will contain the frequencies. In the second part, this output file is used to form the lexicon space and to determine the frequent words that will be used for gathering the bigram statistics. Then the corpus is scanned for the second time to gather the bigram statistics. This process is carried out by the functions named *readfreqs* and *gatherbigrams*. After this point, the categorization process is carried out by *formgroups* and the resulting hierarchy is printed to an output file denoted by the parameter *oflp* by *printgroups*.

A.2.2 Implementation of the Categorization Process

The categorization process is implemented mainly in the function *formgroups*. The specifications and explanations for the functions used in *formgroups* are presented below.

- thetree ∗initializelinksnew()

  Makes a flat linked list using the elements in the lexicon space and returns a pointer to this list. Lexicon space is constructed by using the frequent words. The size of the lexicon space is denoted by the constant *FREQWORDNUM*.

- void initializehasharray(hasharrelm hasharray[FREQWORDNUM])

  Makes necessary initializations on the array that is used for hashing.

- thetree ∗newgreedy(thetree ∗mainlistpointer);

  Applies the modified greedy algorithm on the input embedded-list and returns the new form of the list and a new linked list holding the pointers for the set centers.

- void prestat(thetree *mainlistpointer);

  Takes the embedded list and derives the statistical behavior of each cluster at the highest level of abstraction using fuzzy logic.

Having these procedures to hand the algorithm used by the *formgroups* function could be denoted as follows:

1)Make a flat linked list using the elements in the lexicon space

2)Make necessary initializations.

3)While the embedded has more than one element in the top level

3.1)Apply the modified greedy algorithm on the list.

3.2) Determine the statistical behaviour of newly formed classes which are at the top level of the embedded list.

A.2.3 Implementation example for the distance function

The above functions frequently use the distance function between the linguistic elements. The implementation for this function using the *Combined Metric* is presented below.

```
/* THE DISTANCE FUNCTION BETWEEN TWO LINGUISTIC ELEMENTS */
/* METRIC USED: COMBINED METRIC */
double finddistance(x,y)
int x;
int y;
/* x and y are indexes of the two ling. elm. denoting their location
in the bigram matrix */
{
 int i,tmp;
 double distance=0, dist, u1,d1,d2, distanceeuc;

/* First the cosine of the angle between the vectors produced by the
proceeding words is calculated then the process is carried out on the
vectors formed by preceeding words. So the distance is set to 2 and the
two cosine values are subtracted from this value */

 distance=2;
```

```
/* this part  calculates the cosine of the angle between two
vectors */
/* The cosine given by the proceeding words */
 u1=0;d1=0;d2=0;
 for (i=0; i<WORDNUM; i++)
   u1=u1+(matrix[x][i]*matrix[y][i]);
 for (i=0; i<WORDNUM; i++)
   d1=d1+pow(matrix[x][i],2);
 d1=sqrt(d1);
 for (i=0; i<WORDNUM; i++)
   d2=d2+pow(matrix[y][i],2);
 d2=sqrt(d2);

 dist=u1/(d1*d2);

 distance=distance-dist;



/* The cosine given by the preceeding words */
 u1=0;d1=0;d2=0;
 for (i=0; i<WORDNUM; i++)
   u1=u1+(matrix[i][x]*matrix[i][y]);
 for (i=0; i<WORDNUM; i++)
   d1=d1+pow(matrix[i][x],2);
 d1=sqrt(d1);
 for (i=0; i<WORDNUM; i++)
   d2=d2+pow(matrix[i][y],2);
 d2=sqrt(d2);

 dist=u1/(d1*d2);

distance=distance-dist;
```

```
/* The following part calculates the euclidean distance again both on
proceeding and preceeding words. */

d1=0; d2=0;
for (i=0; i<WORDNUM; i++)
   d1=d1+pow((matrix[x][i]-matrix[y][i]),2);
d1=sqrt(d1);

for (i=0; i<WORDNUM; i++)
   d2=d2+ pow((matrix[i][x]-matrix[i][y]),2);
d2=sqrt(d2);

distanceeuc=d1+d2;
/* The two different values resulting from the preeceding and
proceeding words are added*/

distance = distance*distanceeuc; /* The combined distance is found */

return distance;
}
```

APPENDIX B

CLUSTERS FORMED USING THE COMBINED METRIC

In order to represent the cluster hiearachy obtained using the algorithm a notation using the letters of the alphabet is used. The root node of the tree is shown with the letter b. The number of letters in each node represent the level of that node. Each child of a node has its parent name concatenated with one more letter denoting its location. For instance if bbc is a node having three children. The child nodes will be labelled as bbc-b, bbc-c and bbc-c. So the labels of the nodes can give necessary information about the location of them in the hierarchy.

- set b contains : bb bc
- set bb contains : bbb bbc
- set bbb contains : bbbb bbbc
- set bbbb contains : bbbbb bbbbc
- set bbbbb contains : bbbbbb bbbbbc bbbbbd bbbbbe bbbbbf
- set bbbbbb contains : subject matter question
- set bbbbbc contains : corner sight end sound side
- set bbbbbd contains : middle rest
- set bbbbbe contains : line point story

- set bbbbbf contains : direction cause
- set bbbbc contains : bbbbcb bbbbcc
- set bbbbcb contains : bbbbcbb bbbbcbc bbbbcbd bbbbcbe bbbbcbf bbbbcbg
- set bbbbcbb contains : same first
- set bbbbcbc contains : world room house door
- set bbbbcbd contains : next best most whole old other
- set bbbbcbe contains : professor opposite church hall least present once last baby prisoner doctor wind gate village sun country
- set bbbbcbf contains : street town window ground table carriage road right
- set bbbbcbg contains : earth forest garden truth river
- set bbbbcc contains : bbbbccb bbbbccc bbbbccd
- set bbbbccb contains : city light crowd
- set bbbbccc contains : bank steps wall ladies streets fire floor horses darkness path court watch drawing windows scene news fact sick
- set bbbbccd contains : picture case glass
- set bbbc contains : bbbcb bbbcc
- set bbbcb contains : water family children money land
- set bbbcc contains : captain servant book horse meeting situation circumstances summer afternoon evening night morning day future early
- set bbc contains : bbcb bbcc
- set bbcb contains : bbcbb bbcbc
- set bbcbb contains : bbcbbb bbcbbc
- set bbcbbb contains : large new small great very strange certain good fine few little

- set bbcbbc contains : slight man's sudden thousand hundred different
- set bbcbc contains : bbcbcb bbcbcc bbcbcd bbcbce bbcbcf bbcbcg
- set bbcbcb contains : second big dark
- set bbcbcc contains : rich fair secret blue soft cold bright quick frightened surprised plain clear true greater worse better tall dead living wrong
- set bbcbcd contains : chapter page chief real public
- set bbcbce contains : wonderful sad handsome serious beautiful natural terrible painful happy pleasant hard
- set bbcbcf contains : mere heavy
- set bbcbcg contains : peculiar special wild deep fresh low short high strong particular
- set bbcc contains : bbccb bbccc bbccd bbcce bbccf bbccg
- set bbccb contains : expression air
- set bbccc contains : piece tone smile full kind sort sense state part out instead deal
- set bbccd contains : purpose letter note moment word effort opportunity object meaning power means need use way time beginning strength journey service free distance occasion reason
- set bbcce contains : appearance memory presence idea sake
- set bbccf contains : notice cry hold touch influence act account form effect care
- set bbccg contains : months days hours years consciousness condition view force opinion fear knowledge feeling thought think one gentlemen faces moments character nature spirit business work movement step lives heads minutes

  set bc contains : bcb bcc
- set bcb contains : bcbb bcbc
- set bcbb contains : bcbbb bcbbc

- set bcbbb contains : meant ought wanted used enough back began tried turned came
- set bcbbc contains : seems seemed
- set bcbc contains : bcbcb bcbcc
- set bcbcb contains : bcbcbb bcbcbc
- set bcbcbb contains : bcbcbbb bcbcbbc bcbcbbd bcbcbbe bcbcbbf
- set bcbcbbb contains : take make get be
- set bcbcbbc contains : send save pay
- set bcbcbbd contains : enter pass follow carry call give bring tell do let forgive
- set bcbcbbe contains : show ask hear keep help leave find see
- set bcbcbbf contains : marry meet
- set bcbcbc contains : bcbcbcb bcbcbcc
- set bcbcbcb contains : bcbcbcbb bcbcbcbc bcbcbcbd
- set bcbcbcbb contains : come go
- set bcbcbcbc contains : talk speak
- set bcbcbcbd contains : happen stop wait bear write read begin answer return try order regard run turn drive seem
- set bcbcbcc contains : bcbcbccb bcbcbccc bcbcbccd bcbcbcce
- set bcbcbccb contains : stay live
- set bcbcbccc contains : drink die play fight sleep bed
- set bcbcbccd contains : lie change fall stand walk sit
- set bcbcbcce contains : consider forget understand say imagine
- set bcbcc contains : bcbccb bcbccc
- set bcbccb contains : impossible possible necessary

- set bcbccc contains : obliged bound likely able
- set bcc contains : bccb bccc
- set bccb contains : bccbb bccbc
- set bccbb contains : bccbbb bccbbc
- set bccbbb contains : bccbbbb bccbbbc
- set bccbbbb contains : bccbbbbb bccbbbbc bccbbbbd
- set bccbbbbb contains : bad pretty long
- set bccbbbbc contains : common general
- set bccbbbbd contains : calm pale warm simple sweet quiet busy hot angry ill
- set bccbbbc contains : bccbbbcb bccbbbcc
- set bccbbbcb contains : black white poor young less red more
- set bccbbbcc contains : personal human stone green yellow six three four two further longer doubt
- set bccbbc contains : bccbbcb bccbbcc
- set bccbbcb contains : bccbbcbb bccbbcbc
- set bccbbcbb contains : bccbbcbbb bccbbcbbc
- set bccbbcbbb contains : grass trees
- set bccbbcbbc contains : boys girls
- set bccbbcbc contains : bccbbcbcb bccbbcbcc
- set bccbbcbcb contains : party conversation company
- set bccbbcbcc contains : box name body lad boy child
- set bccbbcc contains : bccbbccb bccbbccc bccbbccd bccbbcce
- set bccbbccb contains : things people men

- set bccbbccc contains : others women
- set bccbbccd contains : invisible evil important honest instant hour
- set bccbbcce contains : countess temple god princess
- set bccbc contains : bccbcb bccbcc
- set bccbcb contains : bccbcbb bccbcbc
- set bccbcbb contains : bccbcbbb bccbcbbc bccbcbbd bccbcbbe
- set bccbcbbb contains : hands hand head
- set bccbcbbc contains : lips arms
- set bccbcbbd contains : former mother's brother father's own usual
- set bccbcbbe contains : aunt uncle sister husband's
- set bccbcbc contains : bccbcbcb bccbcbcc bccbcbcd bccbcbce
- set bccbcbcb contains : front spite
- set bccbcbcc contains : conscious aware none fond afraid thinking
- set bccbcbcd contains : ashamed proud
- set bccbcbce contains : fault relations
- set bccbcc contains : bccbccb bccbccc bccbccd
- set bccbccb contains : bccbccbb bccbccbc
- set bccbccbb contains : bccbccbbb bccbccbbc bccbccbbd bccbccbbe bccbccbbf bccbccbbg
- set bccbccbbb contains : wish want mean
- set bccbccbbc contains : comes appeared
- set bccbccbbd contains : duty attention desire turning coming close
- set bccbccbbe contains : listened liked wished wants talked spoke

- set bccbccbbf contains : listening ready trying going
- set bccbccbbg contains : promised anxious determined difficult glad sure
- set bccbccbc contains : bccbccbcb bccbccbcc bccbccbcd bccbccbce bccbccbcf
- set bccbccbcb contains : seen done
- set bccbccbcc contains : died fallen drawn learned written gone
- set bccbccbcd contains : known taken brought given
- set bccbccbce contains : begun happened
- set bccbccbcf contains : spoken expected
- set bccbccc contains : bccbcccb bccbcccc bccbcccd bccbccce bccbcccf bccbcccg bccbccch bccbccci bccbcccj bccbccck
- set bccbcccb contains : looking looked
- set bccbcccc contains : speaking talking
- set bccbcccd contains : asking telling
- set bccbccce contains : delight pride joy interested 'clock interest confidence
- set bccbcccf contains : settled fixed seated lying bent later
- set bccbcccg contains : reasons passion paid respect pity waiting sorry
- set bccbccch contains : plan chance bit lot
- set bccbccci contains : promise visit
- set bccbcccj contains : stared glanced
- set bccbccck contains : laugh glance look
- set bccbccd contains : bccbccdb bccbccdc bccbccdd bccbccde bccbccdf bccbccdg bccbccdh bccbccdi bccbccdj bccbccdk bccbccdl bccbccdm
- set bccbccdb contains : face eyes

- set bccbccdc contains : imagination legs friends feet hair voice eye fingers books clothes
- set bccbccdd contains : father husband wife mother
- set bccbccde contains : shoulders neck pocket hat chair shoulder arm mouth
- set bccbccdf contains : health coat cap anger surprise dress
- set bccbccdg contains : person girl lady woman gentleman man fellow else thing
- set bccbccdh contains : figure manner
- set bccbccdi contains : affairs age speech action marriage questions ideas looks silence society love experience
- set bccbccdj contains : feelings words thoughts
- set bccbccdk contains : ears master study foot
- set bccbccdl contains : music difficulty pleasure beauty happiness pain writing honor death trouble
- set bccbccdm contains : daughter position soul place son dear heart mind life friend
- set bccc contains : bcccb bcccc
- set bcccb contains : bcccbb bcccbc
- set bcccbb contains : bcccbbb bcccbbc
- set bcccbbb contains : got made
- set bcccbbc contains : found heard
- set bcccbc contains : bcccbcb bcccbcc
- set bcccbcb contains : bcccbcbb bcccbcbc
- set bcccbcbb contains : bcccbcbbb bcccbcbbc bcccbcbbd bcccbcbbe bcccbcbbf bcccbcbbg
- set bcccbcbbb contains : from for with in of by

- set bcccbcbbc contains : under into through on at over
- set bcccbcbbd contains : between towards upon against after before like about round off away down up
- set bcccbcbbe contains : across along among
- set bcccbcbbf contains : opening below closed outside above beyond what's
- set bcccbcbbg contains : during near toward beside within around behind gave told took
- set bcccbcbc contains : bcccbcbcb bcccbcbcc
- set bcccbcbcb contains : bcccbcbcbb bcccbcbcbc bcccbcbcbd bcccbcbcbe bcccbcbcbf bcccbcbcbg bcccbcbcbh bcccbcbcbi bcccbcbcbj
- set bcccbcbcbb contains : is was
- set bcccbcbcbc contains : laughing smiling laughed stopped smiled stood shot wonder
- set bcccbcbcbd contains : until whether since while whom because till where when
- set bcccbcbcbe contains : demanded drove started continued shouted returned doesn't couldn't wouldn't repeated added
- set bcccbcbcbf contains : me him
- set bcccbcbcbg contains : here again
- set bcccbcbcbh contains : forth paused yourself together alone lay themselves except
- set bcccbcbcbi contains : shall should may will must would might i
- set bcccbcbcbj contains : won't cannot can can't are didn't don't
- set bcccbcbcc contains : bcccbcbccb bcccbcbccc bcccbcbccd bcccbcbcce bcccbcbccf bcccbcbccg bcccbcbcch bcccbcbcci bcccbcbccj bcccbcbcck
- set bcccbcbccb contains : did could

- set bcccbcbccc contains : knew felt saw
- set bcccbcbccd contains : reached entered opened
- set bcccbcbcce contains : considered understood knows remembered
- set bcccbcbccf contains : wrote makes showed threw
- set bcccbcbccg contains : spent received caught
- set bcccbcbcch contains : touched followed met struck loved
- set bcccbcbcci contains : burst finished dropped sat laid passed lost changed cut broke set put
- set bcccbcbccj contains : noticed observed believed lived
- set bcccbcbcck contains : drew held kept
- set bcccbcc contains : bcccbccb bcccbccc
- set bcccbccb contains : bcccbccbb bcccbccbc
- set bcccbccbb contains : bcccbccbbb bcccbccbbc
- set bcccbccbbb contains : bcccbccbbbb bcccbccbbbc bcccbccbbbd
- set bcccbccbbbb contains : and to
- set bcccbccbbbc contains : answered has had said cried asked
- set bcccbccbbbd contains : exclaimed says called replied left
- set bcccbccbbc contains : bcccbccbbcb bcccbccbbcc bcccbccbbcd bcccbccbbce bcccbccbbcf bcccbccbbcg
- set bcccbccbbcb contains : she he
- set bcccbccbbcc contains : there it
- set bcccbccbbcd contains : anybody everyone nobody everybody everything
- set bcccbccbbce contains : doing saying
- set bcccbccbbcf contains : gold mine late course

- set bcccbccbbcg contains : supper comfort tea wine suffering fortune blood sorrow paper tears laughter shame dinner
- set bcccbccbc contains : bcccbccbcb bcccbccbcc bcccbccbcd
- set bcccbccbcb contains : bcccbccbcbb bcccbccbcbc bcccbccbcbd
- set bcccbccbcbb contains : led carried sent
- set bcccbccbcbc contains : raised shook
- set bcccbccbcbd contains : shut putting holding carrying
- set bcccbccbcc contains : bcccbccbccb bcccbccbccc bcccbccbccd
- set bcccbccbccb contains : half taking getting having making being
- set bcccbccbccc contains : exactly finding hearing watching all leaving seeing giving keeping knowing
- set bcccbccbccd contains : remember hope suppose know
- set bcccbccbcd contains : bcccbccbcdb bcccbccbcdc bcccbccbcdd bcccbccbcde bcccbccbcdf bcccbccbcdg bcccbccbcdh bcccbccbcdi bcccbccbcdj
- set bcccbccbcdb contains : itself herself myself us them himself
- set bcccbccbcdc contains : moved fell ran walked went
- set bcccbccbcdd contains : pray trust please expect decided creature fancy
- set bcccbccbcde contains : past open
- set bcccbccbcdf contains : you've she's there's that's it's he's
- set bcccbccbcdg contains : directly quickly soon far well fast rising quietly straight forward rose slowly
- set bcccbccbcdh contains : scarcely neither broken hardly
- set bcccbccbcdi contains : nearly running moving walking
- set bcccbccbcdj contains : i'll you'll we'll you'd he'll

- set bcccbccc contains : bcccbcccb bcccbcccc
- set bcccbcccb contains : bcccbcccbb bcccbcccbc bcccbcccbd
- set bcccbcccbb contains : la rather quite too so another every whose yes nor than each both why how now who which what without am were but as that have or
- set bcccbcccbc contains : therefore thus merely really simply always never only also immediately become already ever many often much been suddenly sometimes almost still just perhaps indeed presently then especially yet even
- set bcccbcccbd contains : sir one's thy madame miss however besides they're you're whatever yesterday ago i'm does folks grew remained isn't became war candy either times
- set bcccbcccc contains : bcccbccccb bcccbccccc
- set bcccbccccb contains : bcccbccccbb bcccbccccbc bcccbccccbd bcccbccccbe bc-ccbccccbf
- set bcccbccccbb contains : though if
- set bcccbccccbc contains : passing reading
- set bcccbccccbd contains : surely certainly probably
- set bcccbccccbe contains : bless thank
- set bcccbccccbf contains : believe feel
- set bcccbccccc contains : bcccbcccccb bcccbcccccc bcccbcccccd bcccbccccce bcc-cbcccccf
- set bcccbcccccb contains : sitting standing
- set bcccbcccccc contains : filled pleased
- set bcccbcccccd contains : french brooke defarge english seth ned
- set bcccbccccce contains : week minute year
- set bcccbcccccf contains : ease peace length home

- set bcccc contains : bccccb bccccc bccccd bcccce bccccf bccccg
- set bccccb contains : we they you
- set bccccc contains : anything nothing something
- set bccccd contains : twenty ten five several
- set bcccce contains : clearly easily easy married entirely perfectly silent particularly worth
- set bccccf contains : those these our an a this his their the your my her any no some not such its
- set bccccg contains : mrs. mr.