Natural Language in Document Clustering

By

Catherine Chia-i Leung

A thesis submitted to the Faculty of Graduate Studies in partial fulfillment of the requirements for a degree of Master of Science.

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba
Canada

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

Natural Language in Document Clustering

BY

Catherine Chia-i Leung

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

Master of Science

This thesis is dedicated to

Mom, who is always there for me

and

Dad, who chased away all the big bad doggies

Acknowledgements

I would like to acknowledge several people, without whom this work could not have been possible.

First, I would like to thank my supervisor Dr. Dekang Lin who helped me a great deal throughout my academic career. It was while working for him as an undergrad, that I really learned how to program. His guidance over the years has truly enriched my university career.

I would also like to thank my committee members John Anderson and Simon Liao for their feedback on my work.

Thanks also to my uncle CT Leung, who first encouraged me to go into computer science. Was it not for him, I probably would have ended up as a pharmacist somewhere being bored to death counting pills. Thanks also to Ben for being the other little brother and my aunt May for helping me with all those little details.

To my friend Mariko, who was always there, thanks for coming out to my defence even though it really was "geeky computer stuff". Thanks also for listening all these years. This friendship is something I will always cherish.

To the gang, for those wonderful moments that are truly indescribable. A special thanks goes to Steve and Sonia for pointing out the problems with my initial presentation and correcting my grammar. I would also especially like to thank Steve for finally agreeing to turn the car around before we got hit by lightning. Thanks also to Jeff for ...well, being Jeff

Another friend I'd like to thank is Markian, who first sparked my interest in pursuing my graduate work.

Finally I would like to thank my family. I'd like to thank my mom for being the person who I most wanted to be like, my dad for encouraging me to pursue my dreams and Alvin for all those fun times we had. Your help and encouragement over the years made this work possible.

Abstract

Clusters provide a natural organization for document collections and could make the task of retrieving documents more efficient. Most documents are considered by IR systems as simply a bag of words. This work shows how a document could be represented according to the dependency relationships between words in the document. It also shows how part of speech tags could be added to supplement the typical word representation. Different similarity functions and clustering algorithms are also considered. Results from this work showed that the information theoretic definition of similarity provided a better similarity function than traditional dice and cosine functions for ranking documents. The results also showed that using dependency relationships alone produced worse results than using words alone. The use of part of speech tags also did not show significant improvement over conventional use of words.

Table of Contents

Table of Equations

Table of Figures and Tables

Table of Graphs

1 Background

1.1 Introduction

Document retrieval is an aspect of information retrieval (IR) that involves finding relevant text documents to some given query from a collection of documents. These document collections could be very large and contain documents of various types, such as abstracts, news articles and so forth. Finding a document that matches a query could be a very difficult task.

One technique that could improve this process is that of clustering. Clustering is the process of grouping objects together in order to maximize the similarity between objects within the same cluster and minimize the similarity between objects in different clusters. Essentially it is a way of organizing objects in accordance to their similarity to each other.

In document retrieval, clustering could mean either *term clustering* or *document clustering* [Willet, 1988]. Any query or document could be thought of as a collection of terms. In *term clustering*, different terms are clustered based on the documents they co-occur in [Willet, 1988]. Term clustering can have several benefits. One benefit is that it lowers the number of terms needed to represent a document because individual terms get replaced by the cluster to which they belong. This reduction is important when there are large amounts of data and relatively

limited resources [Baker and McCallum, 1998]. The focus of this work however, will be on document clustering.

In *document clustering*, similar documents are grouped together to form clusters of documents. This kind of clustering provides a notion of how these documents should be organized and can provide several benefits. For example, if we feel that some documents are relevant to a query then documents that are similar to the relevant documents are also likely to be relevant. Similarly if a query is highly dissimilar to some document then it is likely that it will also be dissimilar to other documents within this cluster. Thus, clustering can add to the efficiency of retrieval because we no longer have to compare all the documents to a query. Furthermore, clustering can be used to reduce the search space of the original query [Salton, 1971]. If we find a document that is very similar to a query we could simply return all the documents within that cluster. Similarly if we find a document that is very dissimilar to the query then we could ignore the other documents of that cluster.

Others have also noted that clustering could actually be used as the basis for an IR system. This process is called Scatter-Gather [Cutting et al, 1992]. The idea is that the documents are partitioned into some number of clusters (scatter) and a representative description of each cluster is given to the user. The user can then choose those clusters that they are interested in. The documents in the chosen clusters are then collected together (gather) and the process is then repeated with this

smaller subset. For this type of system the clustering process must not only be accurate but also efficient to allow for its online nature.

Document clustering, like most other processes in document retrieval, use very little if any Natural Language Processing (NLP). That is, very little of the information given by the syntax or semantics of the text is used when processing the document. Most documents are treated as a collection of terms. Typically these terms are simply the words or phrases found in the documents. In this case phrases typically refer to multiword terms such as "hot dog".

The focus of this study will be on document clustering. In particular it will look at the role of NLP in document clustering. This work is largely divided up into three parts. The first part deals with the background information for this work. The second part describes the system that is built for this work. The last part looks at the results of the experiments conducted.

1.2 Information Found in Text

Most IR systems treat documents as a collection of words. These systems normally do not look at how these words relate to each other. It is usually simply a matter of whether a word is present and the frequency that it occurs. When people read text, it is not simply a matter of what words are present. Often, how a word is used, and

how it relates to others words are just as important as the word itself. This chapter will look at the kind of information that could be obtained from a piece of text and the fundamental linguistic principles behind the rest of my work. It will also look at how this information could be automatically extracted.

1.2.1 Part of Speech

Consider the following question. Given the sentence:

Gummi will book the hall tonight

Which of the following sentences are more similar to the above sentence.

1) Gummi will read the book tonight

2) Gummi will reserve the hall tonight

When we read these sentences we would immediately say that sentence 2 is more similar to the given sentence. However, most systems would regard these sentences simply as a collection of words. Thus, since both sentences differ from the given sentence by only one word, the similarity between the given sentence and each of the other two sentences would be the same. Clearly this isn't right.

One simple piece of syntactic knowledge that could be added is a part of speech tag. In the above case, even though the word *book* appears in both the given sentence and sentence 1, the two words are not the same. In the given sentence book was a verb

and in sentence 1 book was a noun. Instead of simply looking at each term as a word we can look at it as a word/part-of-speech pair.

Given Sentence		Sentence 1		Sentence 2	
words	words with part of speech	words	words with part of speech	words	words with part of speech
Gummi	Gummi_N	Gummi	Gummi_N	Gummi	Gummi_N
will	will_V	will	will_V	will	will_V
book	book_V	read	read_V	reserve	reserve_V
the	the_det	the	the_det	the	the_det
hall	hall_N	book	book_N	hall	hall_N
tonight	tonight_P	tonight	tonight_P	tonight	tonight_P
Number of differences:		1	2	1	1

Table 1: Comparison of Word vs. Part of Speech Representation

As we can see from the above table, sentence 2 is indeed more similar to the given sentence when we consider the part of speech tags as well as the words involved. Although this example is admittedly simple and artificial, it still demonstrates the need to disambiguate terms more closely. The use of part of speech tags could be part of the solution.

1.2.2 Dependency Grammar

One way to describe the syntactic structure of a sentence is to use a parse tree. A *dependency tree* is a type of parse tree that describes the syntactic structure of a sentence in terms of a set of dependency relationships. A *dependency relationship* describes how two words in a sentence are related to each other. A *dependency tree* is formed by the dependency relationships needed to describe the sentence. The following is an example of a sentence and the dependency tree for this sentence.

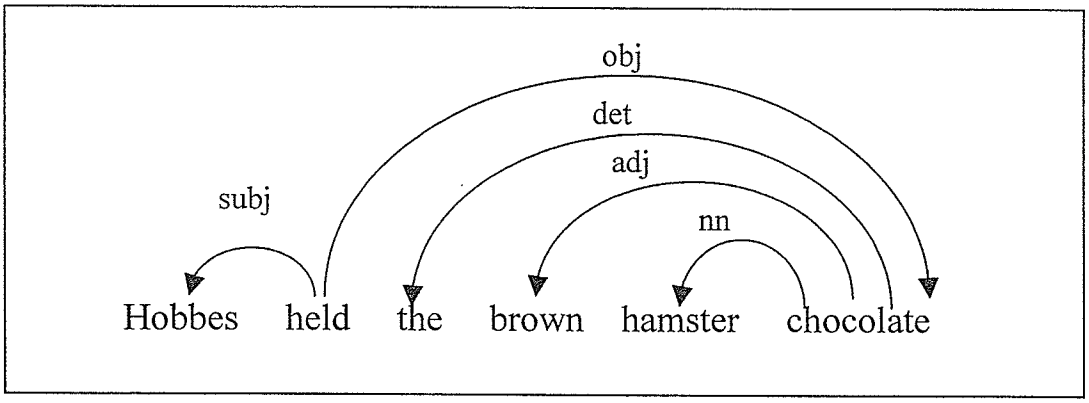


Figure 1: Dependency Tree for "Hobbes held the brown hamster chocolate"

Each arrow represents a dependency relationship and points from the head word to its modifier. Typical dependency grammars allow each word to have at most one head but multiple modifiers [Hudson, 1990]. One property that most head words share is that the modifier changes the head word into a more specific sense of the head word. In the above sentence *hamster* turned *chocolate* into a more specific type

of chocolate, namely a kind of chocolate for a hamster. Similarly brown specifies the colour of the chocolate. If we look at the word *held*, we see that both *Hobbes* and *chocolate* modify *held*. *Hobbes* made *held* more specific in that it specified exactly who did the holding. Similarly *chocolate* made *held* more specific in that it states exactly what was held.

By looking at the syntactic structure in this manner we not only see the syntactic relationships but the semantic relationships as well. That is, not only is the sentence structure considered but its meaning is also taken into account. To observe this, consider the following. In the previous example, 6 words were present: Hobbes, held, the, hamster, brown, and chocolate. Simply knowing that these words were present is not enough because without knowing the syntactic structure, it is very difficult to determine the exact meaning of the sentence. Consider the following:

Given the sentence: ***Hobbes held the brown hamster chocolate***,

Which of the following has a closer meaning to the above sentence?

(a) Hobbes held the chocolate brown hamster.

(b) Hobbes held the white hamster chocolate.

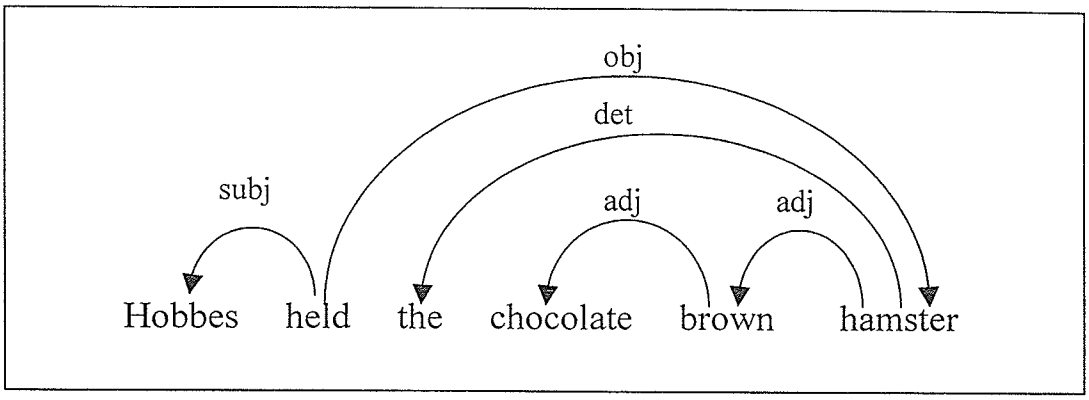


Figure 2:Dependency tree for "Hobbes held the chocolate brown hamster

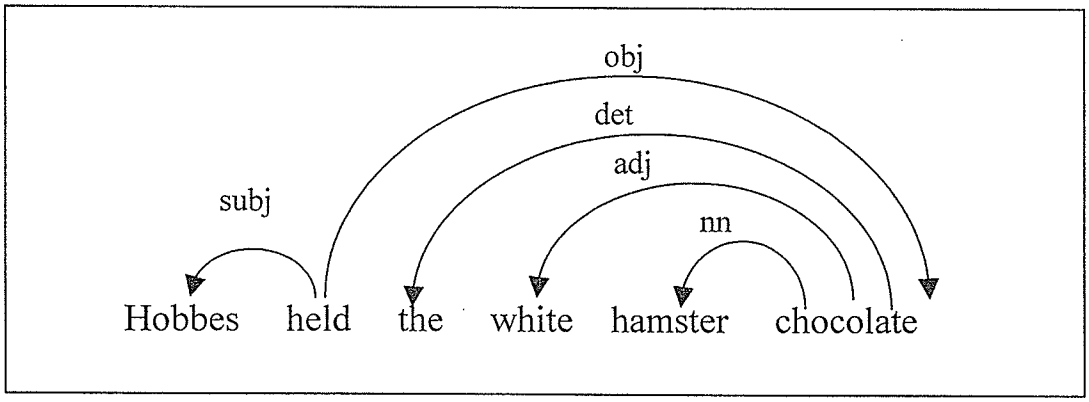


Figure 3: Dependency tree for "Hobbes held the white hamster chocolate"

In terms of meaning it is obvious to us that sentence (b) is much more similar to the original sentence (Hobbes held the brown hamster chocolate) than sentence (a). Typical word representations however would fail to recognize this fact because sentence (a) contains the exact same words as the original sentence whereas sentence (b) has a different word in it. With word representations, sentence (a) would be considered identical to the original sentence. This is clearly not the case. However if we were to consider the head - modifier pairs in each sentence we would find the following.

Original Sentence	Sentence 1	Sentence 2
held – Hobbes	held – Hobbes	held - Hobbes
held – chocolate	held – hamster	held - chocolate
chocolate – the	hamster – the	chocolate - the
chocolate – brown	hamster brown	chocolate - white
chocolate – hamster	brown – chocolate	chocolate - hamster
Number of differences	4	1

Table 2 : Comparison of head-modifier pairs

The reason for using dependency relationships is that these relationships not only state *what* words were in a document but *how* these words were used. Though the relationships are syntactic in nature, the underlying semantic relationship is actually embedded within these syntactic relationships. It is hoped that this information would add extra precision to the task of clustering documents.

1.2.3 MINIPAR

In order to use part of speech tags or dependency relationships, some way of automatically finding these given a piece of natural language text is necessary. To do this MINIPAR was used [Lin]. This section will give a brief description of MINIPAR. The description of how it was actually used in the system will be discussed in a later chapter of this work.

MINIPAR is a broad coverage parser. In other words it is able to handle text about a wide variety of subjects. MINIPAR is the successor to PRINCIPAR [Lin, 1993]. The output and usage of the two parsers are the same but the internal implementation is different.

Like PRINCIPAR, MINIPAR differs from most other parsers in that it does not use a rule-based grammar. Rule-based grammar use many rules to describe how different syntactic structures could be put together to form proper sentences. These rules may describe patterns such as S->NP VP (A sentence is made of a noun phrase followed by a verb phrase). Other rules would then go on to describe what patterns correspond to a Noun Phrase or Verb Phrase. Instead of having all these rules, MINIPAR is based on a grammar that uses a small number of general principles that explains why the language patterns exist.

1.3 Document Representation and Similarity

No matter what clustering algorithm is used, a method of calculating the similarity will be necessary. This chapter will review how documents are typically represented and how similarity could be measured.

1.3.1 Vector Representation and similarity measures

A common way to represent documents in IR involves using a *feature value vector*. Similarity between two documents is calculated by the similarity between its vectors. Each feature in the vector corresponds to a *term* in the document. Each term usually refers simply to a word found in the document. However, terms are not restricted to simply words. Later I will show how syntactic relationships and words can be combined to form terms.

In a feature value vector, the value of each feature could be binary, indicating the presence or absence of a term, or it could be numeric, representing some weight of that term. One possible weight could be the frequency of that term in the document.

Most IR systems employ a stopword list to prevent very common terms such as "the", "of", and "to", from being used. This is done because these words are too common to be representative of a document [Frakes and Baeza-Yates, 1992]. Furthermore stemming algorithms are also commonly used so that different morphological forms of the same term are treated as the one term. For example, "pruning", "pruned", and "prune" could all be treated as the same term.

To calculate the similarity between two documents, their feature value vectors are evaluated according to some similarity coefficient. There are many similarity coefficients that have been used for clustering. For this work I will concentrate on

three coefficients. These include the dice coefficient [Frakes and Baeza-Yates, 1992], the cosine coefficient [Salton and McGill,1983], and a similarity measure based on information theory [Lin,.1998].

Before discussing these coefficients it is necessary to define some variables. The following variables definitions will be used when describing the similarity coefficients.

Let:

D_x represent the vector of terms that represent document x

T(D) represent the set of all unique terms in document D

Weight(t,D) represents the weight of the term t in document D

P(t) represent the probability of the term t being found in any document

The similarity between two documents using the dice similarity measure is:

$$Sim_{dice}(D_1,D_2)=\frac{2\times \sum_{t\in T(D_1)\cap T(D_2)} Weight(t,D_1)\times Weight(t,D_2)}{\sum_{t\in T(D_1)}(Weight(t,D_1))^2+\sum_{t\in T(D_2)}(Weight(t,D_2))^2}$$

[Frakes and Baeza-Yates, 1992]

Equation 1 : Dice Similarity Function

The cosine coefficient is calculated in the following manner:

$$Sim_{\cos}(D_1, D_2) = \frac{\sum_{t \in T(D_1) \cap T(D_2)} Weight(t, D_1) \times Weight(t, D_2)}{\sqrt{\sum_{t \in T(D_1)} (Weight(t, D_1))^2 \times \sum_{t \in T(D_2)} (Weight(t, D_2))^2}}$$

[Salton, and McGill, 1983]

Equation 2:Cosine Similarity Function

Both the dice and cosine coefficient are widely used in IR. They also have the property that the calculated similarity value will be a number between 0 and 1. If two documents share no common term then the similarity is 0. Identical documents would obviously have a similarity of 1.

Another definition of similarity is the *information theoretic definition of similarity.* [Lin, 1998] This definition states that:

$$Sim_{\inf o}(D_1, D_2) = \frac{I(Common\ (D_1, D_2))}{I(Describe\ (D_1, D_2))} = \frac{\log P(Common\ (D_1, D_2))}{\log P(Describe\ (D_1, D_2))}$$

Where:

$I(Common(D_1, D_2))$ is the amount of information given by the commonality between D_1 *and* D_2

$I(Describe(D_1, D_2))$ is the amount of information needed to fully describe D_1 *and* D_2

Equation 3: Information theoretic Definition of Similarity

By the above definition we can calculate *I (Common (D_1, D_2))* and *I(Describe(D_1,D_2))* in the following manner:

$$I(Common(D_1, D_2)) = 2 \sum_{t \in T(D_1) \cap T(D_2)} Min(Weight(t, D_1), Weight(t, D_2)) * (-\log(P(t)))$$

Equation 4: I(Common(D_1,D_2))

$$I(Describe(D_1, D_2)) = \left(\sum_{t \in T(D_1)} Weight(t, D_1) * (-\log(P(t))) \right) + \left(\sum_{t \in T(D_2)} Weight(t, D_2) * (-\log(P(t))) \right)$$

Equation 5:I(Describe(D_1,D_2))

Thus,

$$Sim_{\inf o} = \frac{2 \sum_{t \in T(D_1) \cap T(D_2)} Min(Weight(t, D_1), Weight(t, D_2)) * (-\log(P(t)))}{\left(\sum_{t \in T(D_1)} Weight(t, D_1) * (-\log(P(t))) \right) + \left(\sum_{t \in T(D_2)} Weight(t, D_2) * (-\log(P(t))) \right)}$$

Equation 6: Information Theoretic Definition of Similarity for comparing documents

Clearly in order to use Sim_{info}, it would be necessary to have a way of estimating the probability that a term t will be found in a document. To do this a large corpus is used. Let f (t) represent the number of times the term t appeared in the corpus. Let N be the sum of the frequencies of all terms in the corpus. Then:

$$P(t) = \frac{f(t)}{N}$$

Equation 7:Probability that term t will occur in a document

1.3.1.1 Example:

In the following example I will illustrate how two documents are turned into feature value vectors using a simple word representation and how each similarity measure will calculate their similarity to each other:

My dwarf hamster ate a small, small, carrot.

Document 1

Captain Carrot is a six feet tall dwarf

Document 2

The feature value vectors for these documents would be. The terms that are present in both documents are identified in bold:

a	**carrot**	**dwarf**	ate	hamster	my	small
1	1	1	1	1	1	2

Feature value vector for document 1

a	captain	**carrot**	**dwarf**	feet	is	six	tall
1	1	1	1	1	1	1	1

Feature value vector for document 2

$$Sim_{dice}(D_1, D_2) = \frac{2 \times \sum_{t \in T(D_1) \cap T(D_2)} Weight(t, D_1) \times Weight(t, D_2)}{\sum_{t \in T(D_1)} (Weight(t, D_1))^2 + \sum_{t \in T(D_2)} (Weight(t, D_2))^2}$$

$$= \frac{2 \times ((1 \times 1) + (1 \times 1) + (1 \times 1))}{(1+1+1+1+1+1+4) + (1+1+1+1+1+1+1+1)}$$

$$= \frac{6}{10+8}$$

$$= 0.3333$$

$$Sim_{\cos}(D_1, D_2) = \frac{\sum_{t \in T(D_1) \cap T(D_2)} Weight(t, D_1) \times Weight(t, D_2)}{\sqrt{\sum_{t \in T(D_1)} (Weight(t, D_1))^2 \times \sum_{t \in T(D_2)} (Weight(t, D_2))^2}}$$

$$= \frac{(1 \times 1) + (1 \times 1) + (1 \times 1)}{\sqrt{(1+1+1+1+1+1+4) \times (1+1+1+1+1+1+1+1)}}$$

$$= \frac{3}{\sqrt{10 \times 8}}$$

$$= 0.335$$

$$Sim_{info}(D_1,D_2)=\frac{I(Common(D_1,D_2))}{I(Describe(D_1,D_2))}$$

$$=\frac{\log P(Common(D_1,D_2))}{\log P(Describe(D_1,D_2))}$$

$$=\frac{2\sum_{t\in T(D_1)\cap T(D_2)} Min(Weight(t,D_1),Weight(t,D_2))*(-\log(P(t)))}{\left(\sum_{t\in T(D_1)} Weight(t,D_1)*(-\log(P(t)))\right)+\left(\sum_{t\in T(D_2)} Weight(t,D_2)*(-\log(P(t)))\right)}$$

$$=\frac{2(3.8+11.9+12.1)}{\left(\left(\sum_{t\in T(D_1)}(3.8+11.9+12.1+9.2+14.0+4.5+7.8\right)+\left(\sum_{t\in T(D_2)}(3.8+10.8+11.9+12.1+8.8+4.0+13.2+10.7)\right)\right)}$$

1.4 Document Clustering in IR

This section looks at the role of document clustering in IR. It is a general overview of the approaches and methodologies used in document clustering.

1.4.1 Document clustering algorithms

Document clustering algorithms are typically classified as either hierarchical or non-hierarchical. Hierarchical clustering algorithms organize the documents into a tree like structure. Non-hierarchical algorithms simply partition the documents into

smaller subgroups without forming a structure for them. This section will look at these two main classes for document clustering.

1.4.1.1 *Hierarchical Document clustering*

Hierarchical clustering algorithms can either be divisive or agglomerative. Divisive algorithms start with one large cluster to which all objects belong. This cluster is then divided into two clusters. The process then continues with these new clusters to form smaller and smaller clusters. This kind of hierarchical clustering algorithm is less useful for document clustering because typically each document must contain certain terms to belong to a cluster [Willet, 1988]. The agglomerative clustering algorithms work in the opposite manner. These algorithms begin with each document in its own cluster. These clusters are then repeatedly joined, thereby creating a tree structure. Typically, these algorithms require an $N \times N$ similarity matrix, where N is the number of objects to be clustered. The values in the matrix represent the similarities between each pair of objects. The complexity of calculating the similarity values for this matrix is $O(N^2)$

Different agglomerative clustering algorithms select the clusters to be joined differently. Agglomerative algorithms that have been studied extensively over the years include single link, complete link, group average, and Ward's method [Willet, 1988]. The group average method forms clusters so that the average similarity between a document and all other documents in its cluster is greater than the

document's average similarity with all documents in the other clusters. Ward's method forms clusters by joining those clusters that would result in the lowest increase in the sum of the distance of the documents in a cluster and its centroid. However, for this study I have chosen to use only the single link and complete link agglomerative algorithms.

Single link forms clusters so that each member in a cluster is more similar to at least one other member of that cluster than to any member of another cluster. Complete link forms clusters so that any member of a cluster is more similar to the least similar member of that cluster than to the least similar member of any other cluster [Willet, 1988]. It should be noted that there are actually several specific implementations of each of these algorithms. In general the run time of O (N^2) for single link and complete link may be unsuitable for large collections of documents.

The following is an example, which shows how documents are clustered using both single link and complete link. Each of the matrices shows the similarity between each cluster. Note that once a cluster joins another cluster, it is considered to be a new cluster and thus the individual similarities between other clusters are replaced with a new similarity representing the similarity between the new cluster and the other clusters. The tree structure shows what the clusters look like. Each shape represents a different cluster. Observe how the matrix changes in each case.

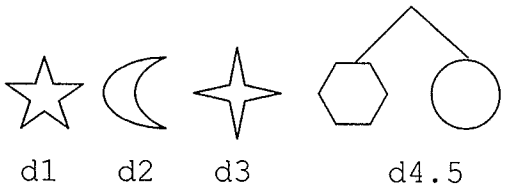
Figure 4 :Comparison of Single Link and Complete Link Algorithms

Single Link

Initial:

	d1	d2	d3	d4	d5
d1	1	.25	.7	.35	.3
d2	.25	1	.15	.2	.5
d3	.7	.15	1	.3	.1
d4	.35	.2	.3	1	.8
d5	.3	.5	.1	.8	1

Most similar are d4 and d5



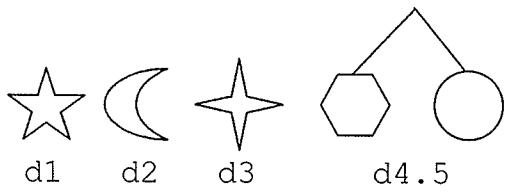
	d1	d2	d3	d4.5
d1	1	.2	.7	**.35**
d2	.2	1	.15	**.5**
d3	.7	.15	1	**.3**
d4.5	**.35**	**.5**	**.3**	**1**

Complete

Initial:

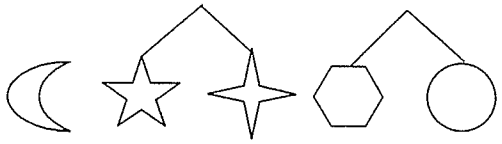
	d1	d2	d3	d4	d5
d1	1	.25	.7	.35	.3
d2	.25	1	.15	.2	.5
d3	.7	.15	1	.3	.1
d4	.35	.2	.3	1	.8
d5	.3	.5	.1	.8	1

Most similar are d4 and d5.



	d1	d2	d3	d4.5
d1	1	.2	.7	**.3**
d2	.2	1	.15	**.2**
d3	.7	.15	1	**.1**
d4.5	**.3**	**.2**	**.1**	**1**

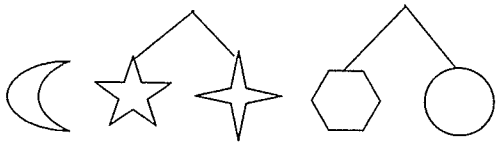
Most similar are d1 and d3



d2 d1.3 d4.5

	d1.3	d2	d4.5
d1.3	**1**	**.2**	**.35**
d2	**.2**	1	..5
d4.5	**.35**	.5	1

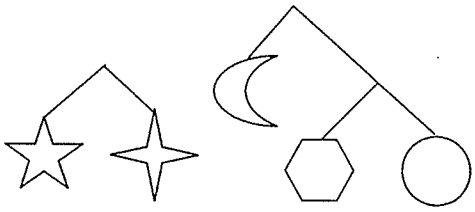
Most similar are d1 and d3.



d2 d1.3 d4.5

	d1.3	d2	d4.5
d1.3	**1**	**.15**	**.2**
d2	**.15**	1	.1
d4.5	**.1**	.2	1

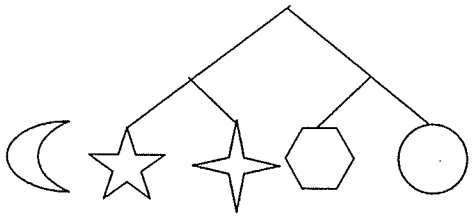
Most similar are d2 and d4.5



d1.3 d2.4.5

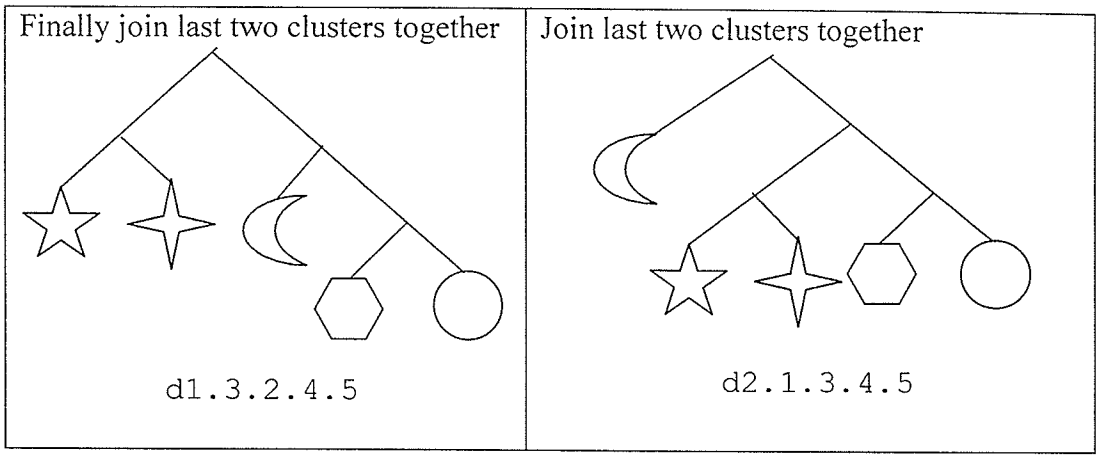
	d1.3	d2.4.5
d1.3	1	**.35**
d2.4.5	**.35**	**1**

Most similar are d1.3 and d4.5



d2 d1.3.4.5

	d1.3.4.5	d2
d1.3.4.5	**1**	**.15**
d2	**.15**	1



1.4.1.2 *Non Hierarchical Document clustering*

Non - hierarchical document clustering assigns the documents into different partitions without any other structuring. They are typically computationally less expensive than hierarchical algorithms with run times at around O(MN) where M is the number of partitions and N is the number of documents. Non-hierarchical clustering was used in most of the earlier work with document clustering due to their inexpensive nature [Frakes and Baeza-Yates, 1992]. However with the increase of computational speed, hierarchical methods have become more popular.

Two general approaches to non-hierarchical clustering are single pass and reallocation methods. Single pass algorithms usually involve choosing some centroid or representative document for each cluster and assign the other documents to these clusters according to its similarity with the representative. A general single pass algorithm works as follows:

- Assign the first document to be the representative of the first cluster

- For every other document:
    - Find the similarity between a document and every existing cluster, keeping track of the cluster with the highest similarity
    - If the highest similarity value exceeds some threshold then add document to that cluster and evaluate the cluster for a new cluster representative
    - Otherwise assign the document as the representative of a new cluster

[Frakes and Baeza-Yates, 1992]

Another approach to non-hierarchical clustering is the reallocation approach. The basic idea here is that, after forming an initial set of clusters, the documents are then re-evaluated to form better cluster. A general approach to this is to first assign documents, by some method, to an initial set of M clusters. A new centroid of each cluster is then chosen and all the documents are re-clustered. This process continues until the new set of clusters is identical to the previous set of clusters.

It should be noted that the above approaches are general descriptions for a type of algorithm. Most non-hierarchical algorithms allow a document to belong to exactly one cluster. There are however, exceptions to this. For example Rocchio's algorithm [Salton, 1971] is a non-hierarchical clustering algorithm that allows documents to belong to multiple clusters or no cluster at all.

1.4.1.3 Non-hierarchical vs. hierarchical

Some clustering algorithms do not fall neatly into the two classes of hierarchical or non-hierarchical clustering. Two such methods are fractionation and buckshot clustering [Cutting et. al, 1992]. The basic idea behind both of these algorithms is to use hierarchical clustering over subset(s) of the document collection. Multiple hierarchies may be formed but because the number of documents used to form any given hierarchy is relatively small, the N^2 runtime is not as big a factor. These hierarchies are then used to find the centres for a set of partitions. All documents are then clustered in a non-hierarchically with these centres that were found.

Although non-hierarchical algorithms are computationally less expensive than hierarchical algorithms, they are less popular. One criticism of non-hierarchical methods is that the clusters they form are dependent on the order with which the documents were presented. In [Cutting et. al, 1992], the authors suggests that earlier work, which used partition clustering to optimize nearest neighbour searching, required many small partitions to be form and that this view made them less attractive despite their low cost.

1.4.2 Scatter Gather

Most of what has been discussed so far looks at clustering as a sort of add on to an existing IR system. However it has been suggested that clustering could actually be used as an information access tool in its own right [Cutting et al, 1992].

Most IR systems are *search* systems. That is the system is given a query and returns a ranked list of documents that match this query. However, another method of looking for information is to *browse* the collection. In browsing all documents in the collection are divided up into smaller subsets. The general content of each subset is shown and the user could choose which subsets they want to look at. A Scatter Gather system is based on this principle of browsing.

In a scatter gather system all the documents in the collection are divided up into clusters (scattered). A general topic description for each cluster is shown. To find documents on a subject, one or more of the clusters are chosen. The documents in those clusters that are picked are then gathered together. These are then re-clustered to form new, more specific clusters. The process is repeated until the desired document(s) are found.

Any scatter gather system requires a fast clustering algorithm. This is because the system is expected to cluster and re-cluster documents on the fly. The typical

approach of hierarchical clustering with its N^2 runtime is not efficient enough for such purposes. For collections of significant size, such as the TREC collection, even the linear runtimes of non-hierarchical algorithms are too slow [Cutting et al, 1993].

Another approach to clustering involves first building a cluster hierarchy [Cutting et. al, 1993] for the collection with standard hierarchical clustering techniques. Every leaf in the hierarchy represents a single document while every internal node represents a meta-document which is like a summary of the its children. The basic idea is that clustering is always done on some constant number of nodes. Thus the interaction time is constant.

1.4.3 Uses of clustering

One of the first uses of clustering was to organize data and help limit the size of the search space [Salton, 1971]. The search proceeded by comparing the query with the centroid vector of each partition. The clusters that exhibited the highest correlation with the query or had a correlation above some threshold were then searched.

Scatter/gather as a stand alone application tool was shown to be less effective than searching techniques [Pirolli et. al, 1996]. However, the combination of scatter/gather browsing with searching has been shown to be more effective than searching alone [Hearst and Pedersen, 1996]. The idea is to use standard searching

techniques to find the documents to be scatter/gathered as oppose to scatter/gathering the entire document collection. A modification to the use of the document hierarchy described in the previous section allowed a near constant runtime for this process [Silverstein and Pedersen, 1997]. This idea of using clustering to better organize retrieval results is very appealing. Studies have shown, that users typically would rather browse through large sets of documents rather than reformulate their query, thus, the use of scatter/gather to organize this data into more manageable components make sense intuitively.

2 System Description

2.1 Introduction

This chapter describes how the experiments were carried out. It begins with a look at the document collection used for the experiment and how each document is processed. An overview of how the program for the clustering experiment is presented next. This is followed by a description of how the different document representations are implemented and how the probabilities for the terms are calculated. The fifth section of this chapter looks at the implementation of the different document clustering algorithms. Finally, the methods of evaluating the clusters are presented. All discussion of experimental results is presented in the next chapter.

2.2 The documents

In order to test how well clustering works, a large collection of documents were needed. To do this part of the document set from TREC was used. TREC is a yearly event sponsored by NIST in the United States. Every year there are a number of tasks that participants can participate in. One of these tasks is the filtering task. For this task, the set of queries or topics is known right from the start but new documents are added to the collection. The task is to decide whether a new document is relevant to a query.

The documents and query are from the TREC-5 filtering task. 48 queries were used in total. Each query had around 1000 documents where relevance judgements were available. In other words, each document had been judged to be either relevant or irrelevant to a given query. Documents sizes varied widely. Size could range from 100 bytes to 10 kb+.

The size of each query and documents varied widely. However, queries were typically small compared to the documents. All documents and queries were in plain text but often contained tags that indicated different sections of the document such as title, description, text, and so on.

2.2.1 The life of a document or query

As stated earlier all queries and documents from this collection are plain text files with tags to indicate different sections of the query or document. For the documents, only the text in the <TEXT> section is used. For the queries, there is no TEXT section, thus any text other than that in the <head> and <num> sections are used.

Once these documents and queries have had their tags removed they are parsed by MINIPAR. From each file, three separate files are generated. The first of these files contain a list of terms which are basically list all the word and phrases of the original document. The second file is very similar to the first except that it also contains a

part of speech tag for every word or phrase. The third document contains a list of all the head/modifier pairs and their relationships. The actual clustering program then uses these three documents to perform the clustering

The following example illustrates the three files that are generated when given a single text file. The example is actually generated by minipar, using the same rules as those used for parsing queries and documents.

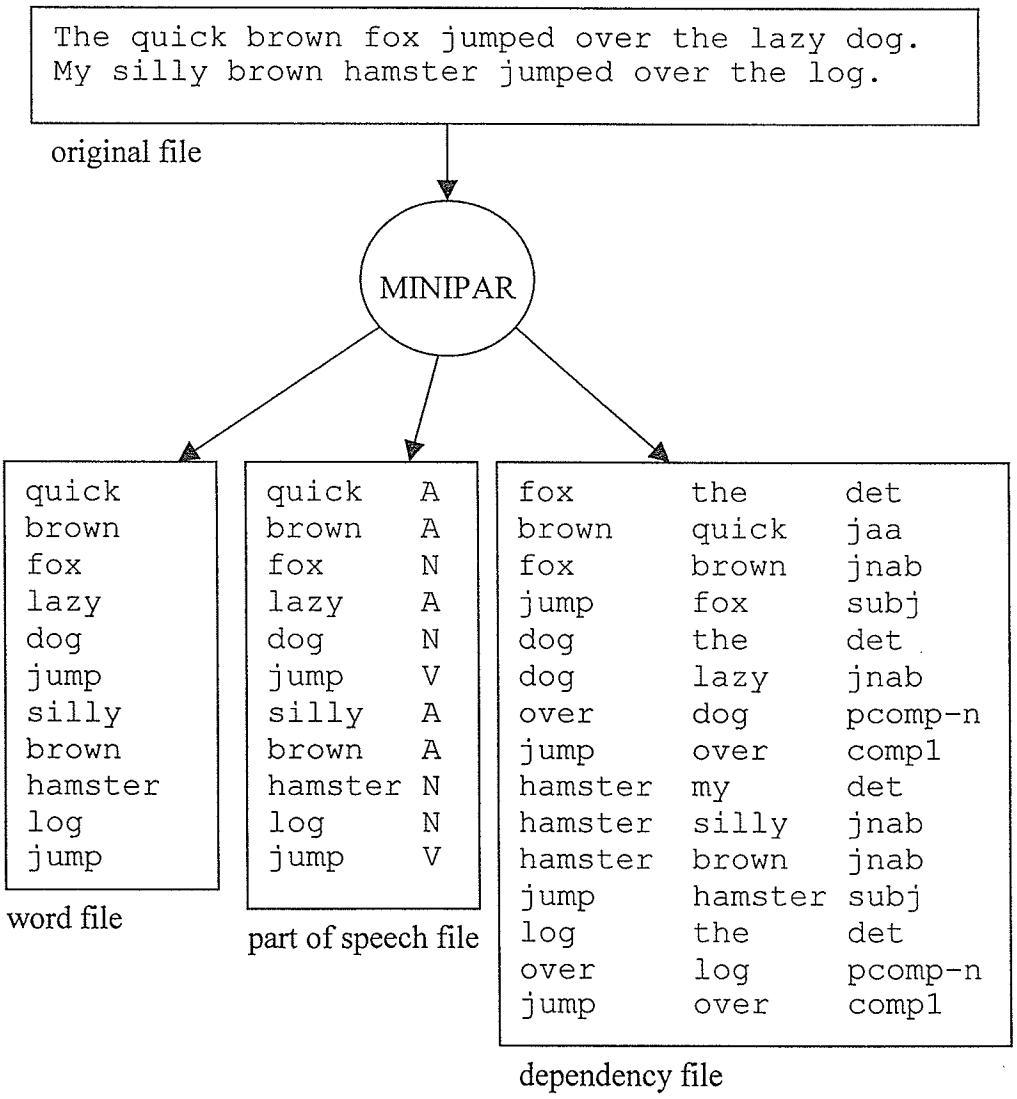


Figure 5:Creation of files for clustering

It should be noted that for all three files, certain words/relationships are not added to the files as they contain little information. For example, the word "the" is not listed, as it is too common to have any meaning. Once these files are created, the three documents are used in separate calls to the clustering program. The next section gives an overview of how the program works.

2.3 Clustering Program overview

Every time the clustering is called, it requires three parameters:

1) The query number
2) The similarity function
3) The representation of the document.

The query number determines the exact set of files to be used. Only those documents where relevance judgements are available for a given query are used in clustering. This is done because, without these judgements it would be hard to evaluate the clustering

The representation of the document could either be word, part of speech, dependency head/modifier pair, dependency head/relationship pair, and dependency

head/modifier/relationship tuple. The representation determines which form of the document will be used.

The similarity function is also given as a parameter because how the documents are compared is affected by the similarity function used. Also, each run of the program could take up to several hours for any one given similarity function. It was a good idea to break it up so that if a machine should crash while the program was running only part of the data would be lost.

Once the documents to use and similarity function is decided, all documents are read into the clustering program. Each document is first compared to the query and a similarity, which is calculated according to the chosen similarity function, between it and the query is obtained. The documents are then ranked according to its similarity to the query.

Once this is accomplished, document clustering begins. Clustering occurs only on the top n documents; where n could be 100, 200, or 300 for hierarchical clustering algorithms, and 100, 200, 300 or all documents for non-hierarchical clustering. This is done because all hierarchical clustering algorithms have a run time of $O(n^2)$ where n is the number of documents clustered. Thus, in order to ensure that clustering does not take too long, a maximum of 300 documents are clustered if the algorithm is hierarchical in nature.

For every clustering algorithm, the documents are clustered multiple times. Each time, either the number of documents clustered is different, or the maximum number of partitions is different.

After clustering, the clusters are sorted by its average similarity to the query. This value is used to order the clusters. The documents that were not clustered are considered to be a junk cluster and was always ranked last. Starting with the most similar cluster, the documents are placed back in a list. Thus any document in a higher cluster always occur in the list before any document in a lower cluster regardless of its original ranking. This new list is evaluated and the results recorded. Furthermore, the three clusters that have the highest number of relevant documents are evaluated on their own.

Thus for every query, we look at all 5 different representations of the query and its documents. Each of these is then clustered according to a different similarity function and the results are recorded. In the next sections we will look closer at how documents were represented and how each clustering algorithm worked.

2.4 Document Representation and Similarity Functions

As stated earlier, part of this work is to evaluate whether or not different document representations would give better clustering results. Furthermore, one of the similarity functions (Sim_{info}) is dependent on the probability of certain events occurring. Thus, how a document is represented affects how the similarity function behaves. This section will look at the different document representations and how Sim_{info} is affected by this.

2.4.1 As Words

This is the simplest representation of a document. We view the document as being made up of a bag of words or phrases. Note that phrases in this case refer to multiword terms such as "hot dog" and not true syntactic phrases. With a word representation, we simply compare documents based on how often a word or phrase occurs within the document. Thus, each term of our feature value vector represents the number of times a specific word or phrase occurred within a document. The following formula for probability that a particular term will be found within a document is used to calculate Sim_{info}

$$P(term\ found\ in\ document) = \frac{Number\ of\ times\ term\ appeared\ in\ N\ documents}{Total\ number\ of\ terms\ in\ N\ documents}$$

Equation 8 : probability of finding a term using word represenation

2.4.2 As Words with Part of Speech Tags

This representation is very similar to the basic word representation. The only difference here is that associated with every word is a part of speech tag that identifies the role of the word within a sentence. Thus a term in this case refer to the word/part of speech pair as opposed to simply the word on its own. The probability of finding a word with a particular part of speech is defined as:

$$P(T_{w,p}) = \frac{Number\ of\ times\ w\ appeared\ as\ a\ p\ in\ N\ documents}{Total\ number\ of\ terms\ appearing\ as\ p\ within\ N\ documents}$$

where Tw,p is a term with the word or phrase w and the part of speech p

Equation 9:Probability of finding a term using Part of Speech Representation

2.4.3 As Dependency Relationships

A dependency relationship has three components: the head word, the modifier and the relationship between the head and the modifier. Part of this work is to determine which of these are useful in clustering. For these experiments, three uses of the dependency relationships were studied. The first is to use all three components. This representation will be referred to from here on as the Dep3 representation. The second is to use only the head word and its relationship type. From here on, this representation will be known as the Depht representation. The third method is to use the head word and the modifier. This third method will be known from here on as the Dephm representation.

The following functions indicate how the probability of a term is calculated for dep3, depht and dephm representations.

$$P(T_{h,t}) = \frac{Number\ of\ times\ h\ relates\ to\ any\ word\ with\ relationship\ type\ t\ in\ N\ documents}{Total\ number\ of\ terms\ in\ N\ documents}$$

$$where\ T_{h,t}\ is\ a\ term\ with\ head\ word\ h\ and\ relationship\ t$$

Equation 10:Probability of finding a term using DepHT representation

$$P(T_{h,t,m}) = \frac{Number\ of\ times\ h\ modifies\ m\ with\ relationship\ type\ t\ in\ N\ documents}{Total\ number\ of\ terms\ in\ N\ documents}$$

$$where\ T_{h,t,m}\ is\ a\ term\ with\ head\ word\ h\ and\ modifier\ m$$

Equation 12:Probability of finding a term using DepHT representation

$$P(T_{h,m}) = \frac{Number\ of\ times\ h\ \text{modifies}\ m\ in\ N\ documents}{Total\ number\ of\ terms\ in\ N\ documents}$$

$$where\ T_{h,m}\ is\ a\ term\ with\ head\ word\ h\ and\ \text{modifier}\ m$$

Equation 11:Probability of finding a term using Dephm representation

2.5 Clustering Algorithms

As stated in the background portions of my work, both single link and complete link have been used earlier in document retrieval. For my work, these two algorithms

were used to provide a basis of comparison. Aside from these, two other clustering algorithms were also looked at. One was a non-hierarchical clustering algorithm and the third was a hybrid of single and complete link algorithms. This section will look at these clustering algorithms in more details.

2.5.1 Single and Complete Link

Single link and complete link are standard clustering algorithms. The description of their general operation was provided in the background section of this work. In this portion, I would like to present the one modification I made to these algorithms.

Recall that single link and complete link cluster documents by iteratively combining the most similar clusters to form a new cluster. As a result, a tree like structure is formed. Each sub-tree can be viewed as a separate cluster. Also recall that the experiments reorder the documents according to which cluster the document belonged to. If we were to form one big tree structure, we would need to break it up so that we would have these separate clusters. Thus, instead of forming one big cluster, the modification made to the single link and complete link algorithms stopped the clustering when the set number of clusters remained. This same modification was made to the hybrid hierarchical clustering algorithm

2.5.2 Hybrid Hierarchical Algorithm

Recall that single link works by clustering the most similar two clusters. Once this new cluster is formed, its similarity to another cluster is the higher of the two original similarities. This sometimes will cause the clustering algorithm to generate long chain-like cluster. Thus, the following hybrid algorithm was introduced to study whether or not it could help alleviate this problem.

This modified version of single link clustering works for the most part just like the single link algorithm. The problem in regular single link was caused by the fact that any high value for similarity between any two members of different clusters became the dominant similarity value. This occurred regardless of how similar other members of each cluster was to each other. Thus, in this modified version of single link, instead of substituting the higher similarity value between any cluster and a newly formed cluster, we look at whether or not the lower similarity value is too low. This was achieved by checking whether or not the lower value was below a certain threshold value. If it was below the threshold then set the similarity to 0. In other words if any member of the cluster was too different to another cluster, they should not be in the same cluster. The threshold value was set at 0.1 during these experiments.

2.5.3 Non-hierarchical clustering

Recall that the general non-hierarchical algorithm assigns documents to clusters by finding the cluster that the document is most similar to. If the most similar document is below some threshold, the document becomes a new cluster. The non-hierarchical clustering algorithm used for this work uses this basic algorithm. The threshold value in this case is not a fixed value but is determined by the differences in the similarity of the document to the cluster.

Furthermore, instead of allowing an unlimited number of clusters to be formed, a maximum number of clusters is used. Any document that would have formed its own cluster after the maximum number of clusters has been formed is placed into a junk cluster. Any document not used for clustering is automatically considered to be part of the junk cluster.

2.6 Evaluation

Two standard measurements of performance in information retrieval are precision and recall. Precision measures how accurate the retrieval results are. Recall measures the percentage of relevant documents returned. For example, suppose that some search engine was given a query, and after searching through a set of documents, returned 20 documents that it considered as matching to the query.

Suppose that of these documents 14 were truly relevant to the query. This would mean that the precision of the system is 70% because 70% of those documents returned match the query. Suppose that within the document collection there were actually 35 documents that had matched the query. Since only 14 of the 35 were returned, its recall would be 40% because only 40% of the relevant documents were retrieved

These two measurements are typically used together on a ranked list of documents. This list represents the order in which the documents would have been retrieved by the system. The best possible list would have all relevant documents ranked before all irrelevant documents. To apply precision and recall measurements to this list, we measure the precision at recall levels, 0%,10%, 20%,...100%. We define the precision at recall level n% as the highest precision measured after seeing at least n% of all relevant documents.

To illustrate this, consider the following list, which represents a ranked list of documents. In the table below, the first row represents whether a document is relevant to a query. An 'R' stands for relevant; an 'I' stands for irrelevant. The second row holds the precision of the document list after seeing that document.

Note that there are a total of 18 documents. Of these 11 are relevant.

R	R	R	R	I	I	R	R	R	I	R	R	R	I	R	I
1	1	1	1	.8	.67	.71	.75	.78	.7	.73	.75	.77	.71	.73	.68
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Since there are 11 relevant documents, after seeing one document we have found 9.1% of our documents. After 2 documents we would have found 18.2% of our documents. Below, the arrows indicate where the boundary for each recall level would be.

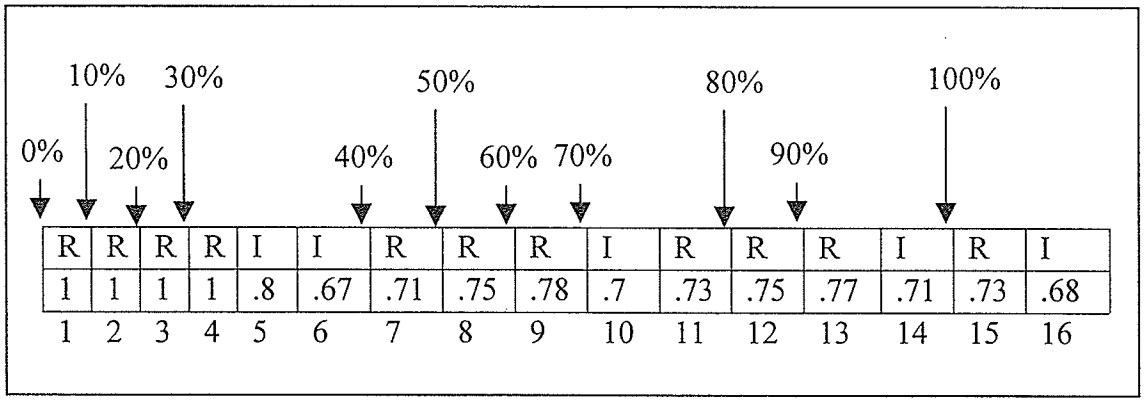


Figure 6 : Recall Level Cutoffs

To properly measure the precision recall find the highest value for precision starting from the boundary to the end of the list. This is the precision for the recall level indicated.

to ultimately improve retrieval results. If clusters were well formed, then, we would be able to improve search results by moving more relevant documents to the beginning of the list because the cluster would be more relevant.

The second part of the evaluation, evaluates the use of clustering as a browsing tool. If clusters were part of a browsing tool, then the user of the system would interactively choose which clusters best matched their needs. If they see that a cluster is relevant to their need then it would be chosen as one of the clusters to look at. Since there were no users it was very difficult to see which clusters would actually be chosen. For estimating how a user might choose which clusters to pick, the three clusters that contained the highest ratio of relevant documents were individually ranked and evaluated. The assumption here is that the user would be able to pick out the cluster that had the most relevant data. If the clusters were well formed then within the chosen clusters, the probability of finding a document that matched the query would be much higher than finding the document outside of the cluster. The same basic precision/recall mechanism is applied to the documents within a cluster. The number of documents and the number of relevant documents within these three clusters are also recorded.

3 Observations and Results

3.1 Results

This chapter will examine the results of the experiments conducted. It will consider how clustering could be best applied as a useful means of information retrieval. Furthermore, it will attempt to determine how different factors such as clustering algorithm, use of natural language representations, and number of documents clustered could affect clustering results.

To facilitate this discussion the following terms will be used throughout this section.

word - word representation

pos - part of speech document representation

dephm - head/modifier dependency representation

depht - head/relationship dependency representation

dep3 - head/relationship/modifier dependency representation

cos - similarity function

dice - dice similarity function

info - information theoretic similarity function

slink - single link clustering algorithm

clink - complete link clustering algorithm

mslink - single link/complete link hybrid clustering algorithm

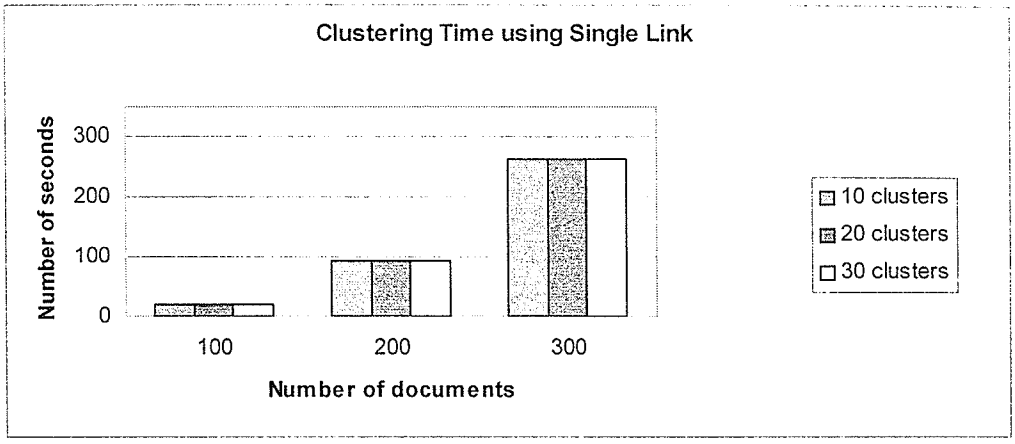
non-hier - non-hierarchical clustering algorithm

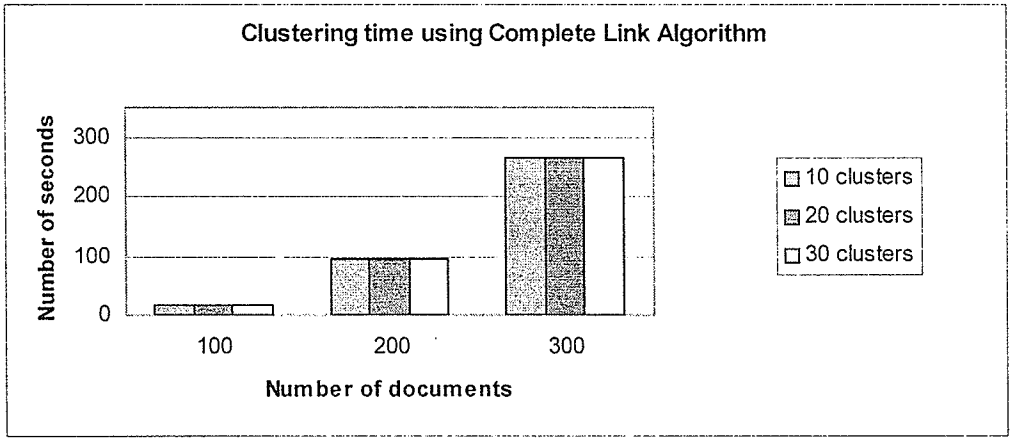
3.2 Efficiency

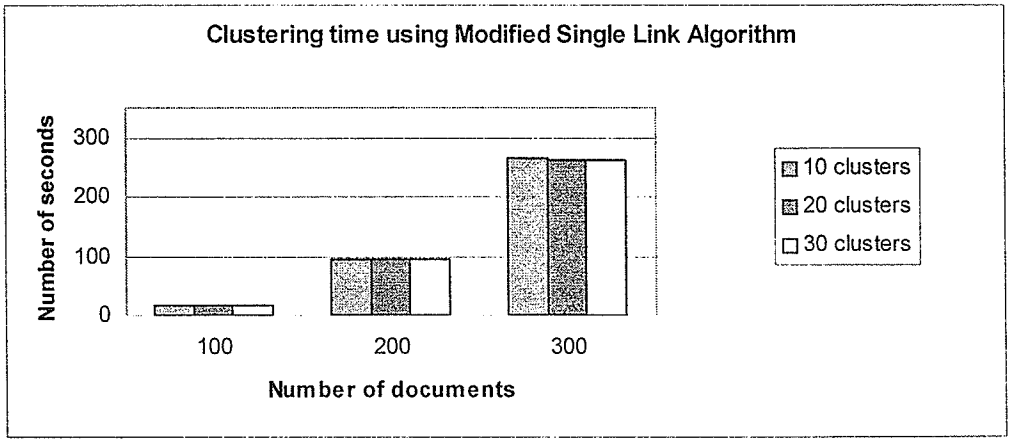
An important consideration when evaluating clustering algorithms is the efficiency of the clustering algorithms. One of the purposes of clustering is to organize retrieved documents. Since the set of retrieved documents is not static, our clustering algorithm must be fast enough to cluster a given set of documents in a reasonable amount of time. This section will look at how different similarity functions and document representations affect the speed of clustering.

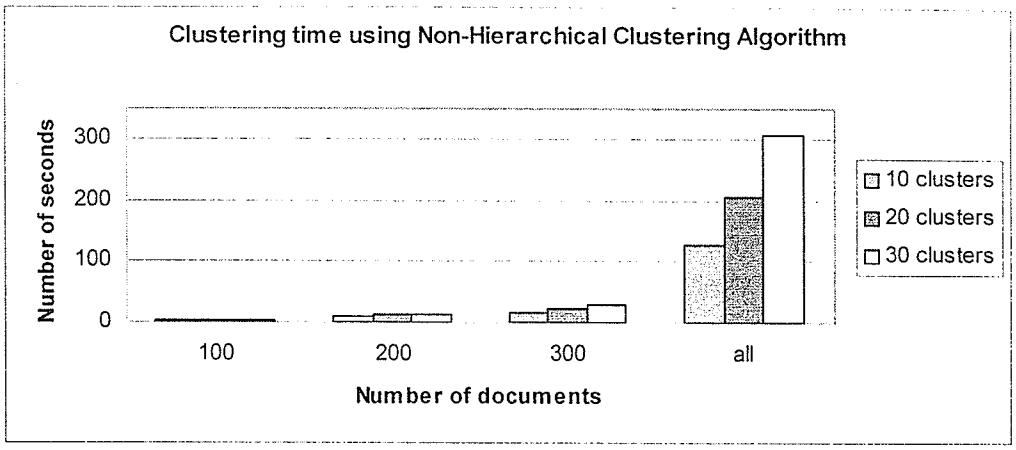
3.2.1 Efficiency of different clustering algorithms.

As stated earlier hierarchical clustering algorithms generate a matrix of similarity values. The runtime clustering algorithms for these algorithms are typically $O(n^2)$ where n is the number of documents clustered. Due to this reason, when using any of the hierarchical clustering algorithms, no more than 300 documents were clustered. Non-hierarchical clustering algorithms tend to have a linear run time so a clustering of all documents was performed. The following charts shows the run times of different clustering algorithms, and how cluster size and the maximum number of documents clusters affect their efficiency.


Clustering Time using Single Link
Number of seconds
300
200
100
0
100
200
300
Number of documents
10 clusters
20 clusters
30 clusters


Clustering time using Complete Link Algorithm
Number of seconds
300
200
100
0
100
200
300
Number of documents
10 clusters
20 clusters
30 clusters


Clustering time using Modified Single Link Algorithm
Number of seconds
300
200
100
0
100
200
300
Number of documents
10 clusters
20 clusters
30 clusters



From the previous charts, we can clearly observe two main points:

1. All hierarchical algorithms take a similar amount of time for clustering the same number of documents.
2. The number of clusters formed has little effect on the speed of clustering for hierarchical algorithms.

However, it should also be noted that, the non-hierarchical clustering algorithm is affected by both the number of documents clustered and by the number of clusters formed. This is reasonable because the non-hierarchical algorithm works by comparing each document with all clusters already formed. Thus, if we allow more clusters to be formed, each document will need to be compared to more clusters.

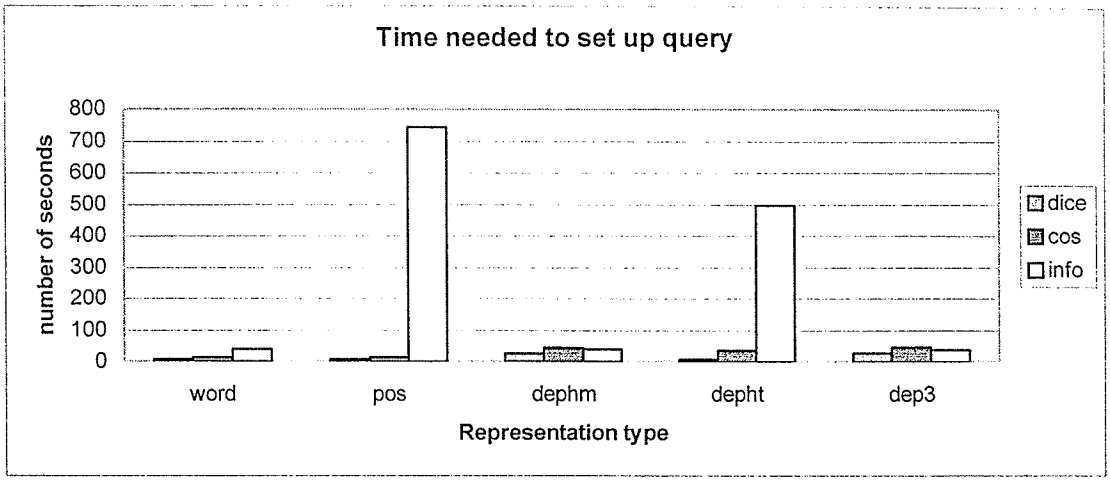
We can also observe that the non-hierarchical clustering algorithm is much faster than the hierarchical algorithms if the same number of documents were clustered. The time it takes the non-hierarchical clustering algorithm to cluster all documents

(around 1092 per query) into 30 clusters is only approximately 50 seconds more than clustering just 300 documents into 30 clusters for hierarchical algorithms. Thus, we see that in practice, non-hierarchical clustering algorithms outperform hierarchical clustering algorithms in terms of efficiency

3.2.2 Efficiency of similarity functions and representations

Although the basic algorithm of comparing two documents is the same regardless of the similarity function and document representation used, it is nevertheless important to consider their effects on the efficiency of clustering. For instance, Sim_{info} requires looking up the probability that a given term will appear in a document. How would this affect the efficiency of clustering? The number of terms used to represent a document will be different for different document representations. Is this difference big enough to significantly or seriously affect performance?

First let us define the term ***clustering time*** as the time needed to cluster ***n*** documents into ***m*** clusters using one particular document representation/similarity function/clustering algorithm combination. The following chart shows the average clustering time of for any combination of similarity function and document representation. The average clustering time below is calculated as the average clustering time for all clustering algorithms using all combinations of n=100, 200, 300 for all documents(non-hierarchical algorithm only) and m=10,20 or 30 clusters.



From the above chart we can see that word and pos representations were the fastest representations when dice or cosine representation was used. This could probably be explained by the fact that there are fewer unique terms for word and part of speech representations than there are for any of the dependency representations. Thus clustering is faster because fewer terms are compared.

When using the information similarity function, the word representation was actually slightly slower on average than the dephm and dep3 representations. Also we should note that the speed of both depht and pos slows dramatically when used in combination with the Sim_{info}. This could actually be explained by the fact that looking up probabilities for each term of dephm and dep3 is a very simple table look up, whereas both pos and depht requires looking up several values and summing up their occurrences before the probabilities could be calculated. Similarly, this explains why the word representation slowed down slightly as well for the sim_{info} function.

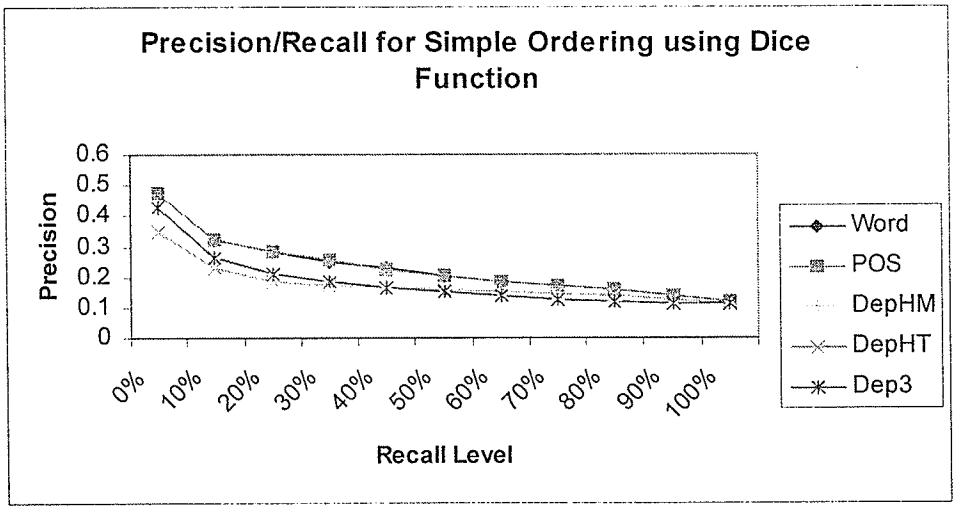
Another observation that can be made is that the dice similarity function is the fastest one no matter what representation was used. Depending on the document representation the info similarity function could be quite slow. However, it should be noted that for the representations where the probabilities could be calculated with a simple table look up, the info similarity function was slightly faster than the cosine representation.

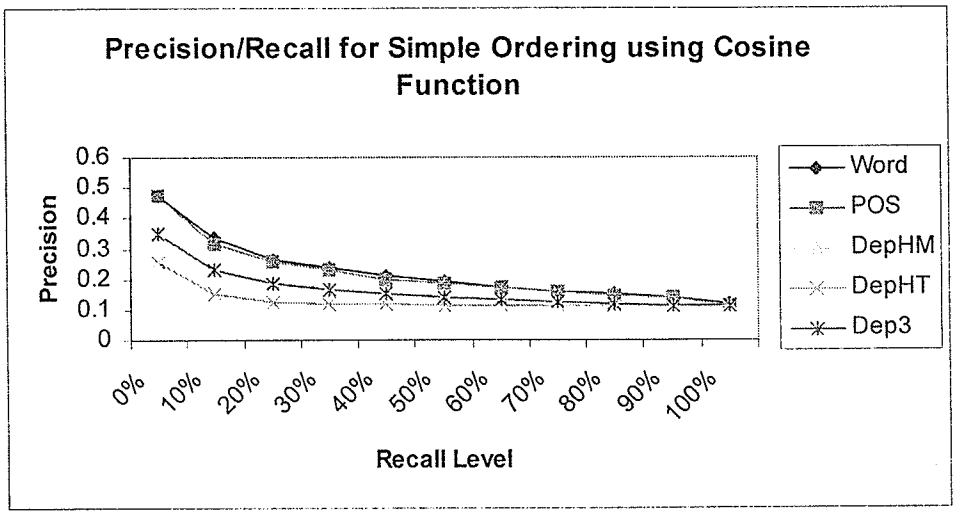
3.3 Precision-Recall ranking

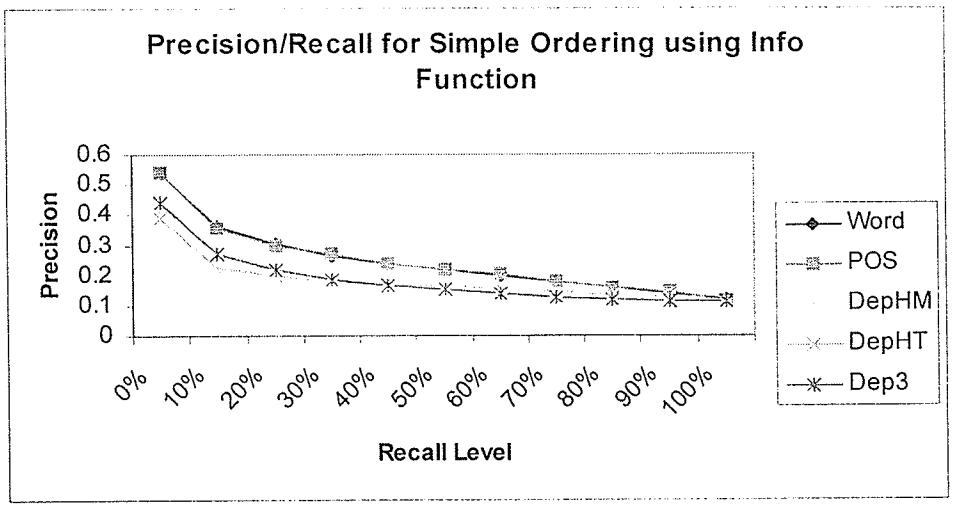
In chapter 2 the standard method of evaluating IR systems was introduced. This method involves measuring the precision of retrieval results at recall levels 0%, 10%, 20%...100%. This section compares the precision recall values of a simple ordering (no clustering is used) to the precision recall values of a list formed using clusters.

3.3.1 Simple ordering without clustering

Before any clustering is performed, all documents are first compared to the query and ordered according to its similarity to the query. The following charts illustrate the precision/recall evaluation of ordering the documents using different document representations and similarity functions.

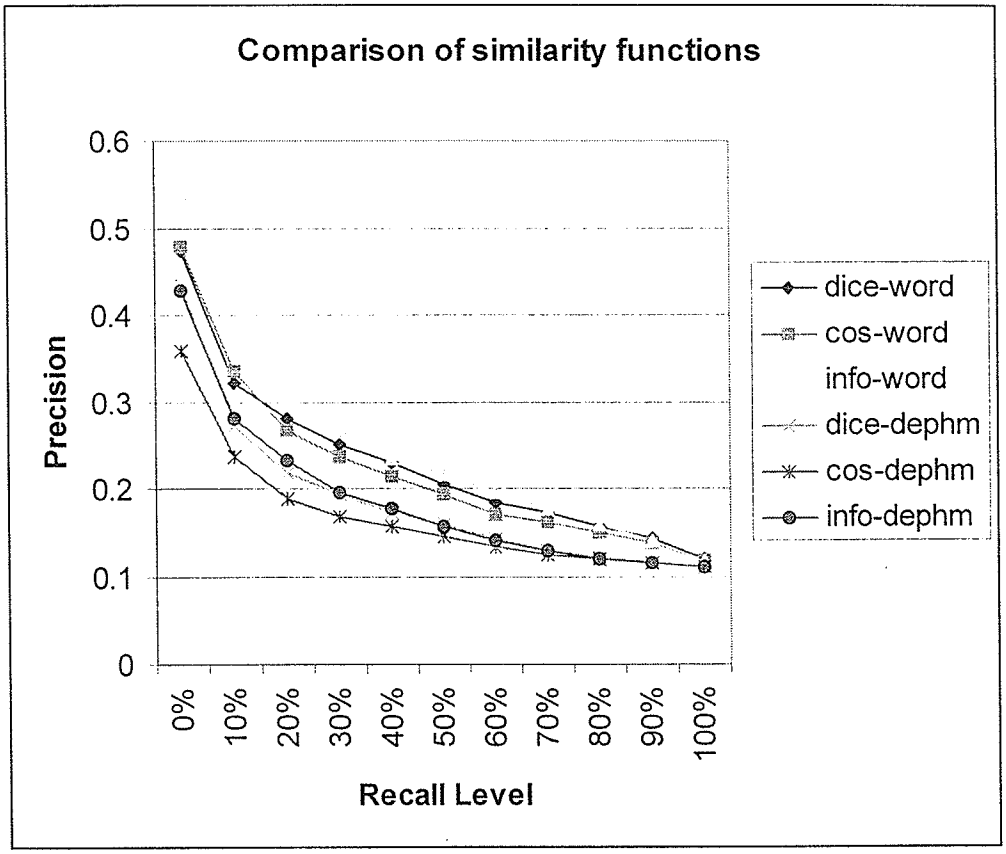






For every similarity function, we can see that word and part of speech representations have the best precision/recall values. Dephm and dep3 representations perform at a similar level, which is always lower than word and part of speech. The worst performing representation is depht.

The other interesting observation is that the best performing similarity function seems to be the info similarity function. This can be more easily observed if we looked at the precision recall values for just word and dephm. Since the pos representation and dep3 representation are similar to word and dephm respectively, we can observe this without their presence in the graph below:

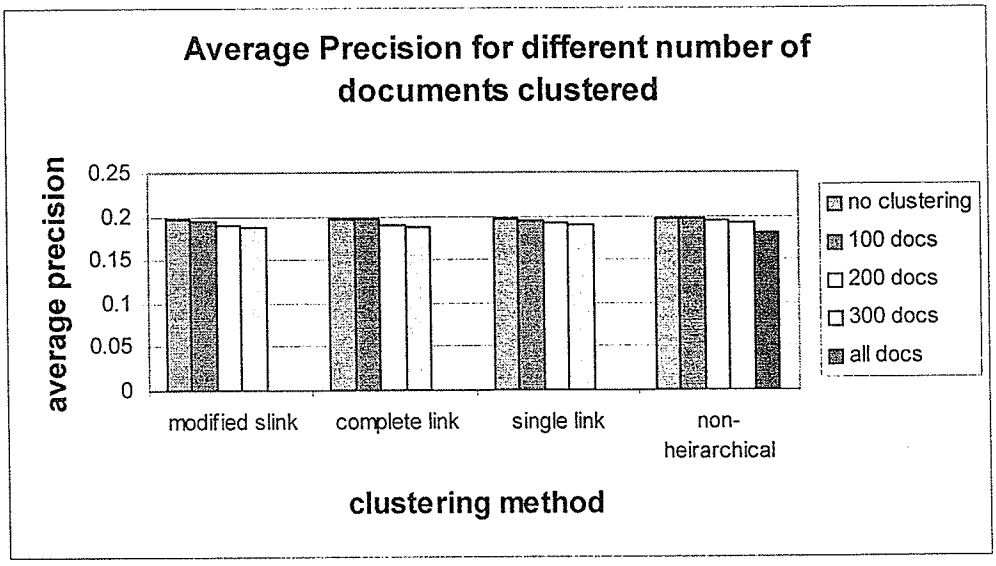


3.3.2 Reordering after clustering

Once the documents are ordered, clustering using the same similarity function and document representation occurs for each of the clustering algorithms. Once the clusters are formed, the clusters are ordered according to its similarity to the query. The documents within each cluster are then placed back into order so that all documents within one cluster are placed before all documents of the next cluster.

The results from simply reordering the documents after clustering were not very good. Clustering results varied widely depending on the number of clusters formed and number of documents clustered. Typically, the reordering did not produce precision/recall values that were significantly better or worse than the original ranking using the same document representation and similarity function. Furthermore, the average precision (the average of the precision over all recall levels) was usually slightly lower than the average precision of the original ordering. This held true for all the different clustering algorithms used.

The average precision of clustering seems most affected by the number of documents clustered. The following chart illustrates the average precision over all document representations and clustering algorithms for different number of documents clustered:



The difference in the average precision is not very high but it does seem to show a general trend that clustering more documents tends to reduce the average precision.

Another observation that can be made from the above chart is that when clustering the same number of documents, the average precision for different number of document representations are fairly similar for each clustering method. Thus it seems that the method of clustering has little effect on the average precision.

3.4 Results of Individual clusters

By using clustering, documents that are similar to a query should fall within the same cluster while documents that are not should be in other clusters. Once all clusters are formed, the three clusters that contain the highest percentage of relevant documents are recorded. This method of selecting which clusters to look at is akin to having a user choose the clusters that best represent their query. This section will examine properties of these clusters and the effects of different document representation and clustering algorithms on them.

There are several factors to consider when studying these clusters. Firstly we should examine the number of relevant documents that were actually clustered. Recall that prior to clustering all documents are first ordered by its similarity to the query. Only the top n documents are actually clustered. We should take into account the fact that

some of the relevant documents may not have been ranked among the top n documents, and thus were not clustered

Furthermore, we have thus far examined performance in terms of precision/recall. However, clustering has more to do with grouping documents so that relevant documents end up in the same cluster and irrelevant documents in different clusters. Thus this section will study those clusters that have the highest proportion of relevant documents.

This measurement of a cluster can be thought of as another measurement of precision. However, instead of measuring the precision for a ranking of documents, we are measuring how precise an entire cluster match a query. By looking at the clusters that have the highest proportion of relevant documents we are basically looking at whether or not clustering was able to more successfully identify the documents that best matched the query and group them together. It should be noted that when ranking clusters for a re-listing of the documents (the results of the previous section), these clusters with a higher proportion of relevant documents may not have been ranked near the top. The intention of the original cluster reordering was to see whether or not we could create a better ranking without any sort of intervention. Since there is no way to actually know the relevance of a document, the heuristic used to rank clusters was based on the similarity between the query and members of the cluster. Thus, the reordering of documents based on the clusters

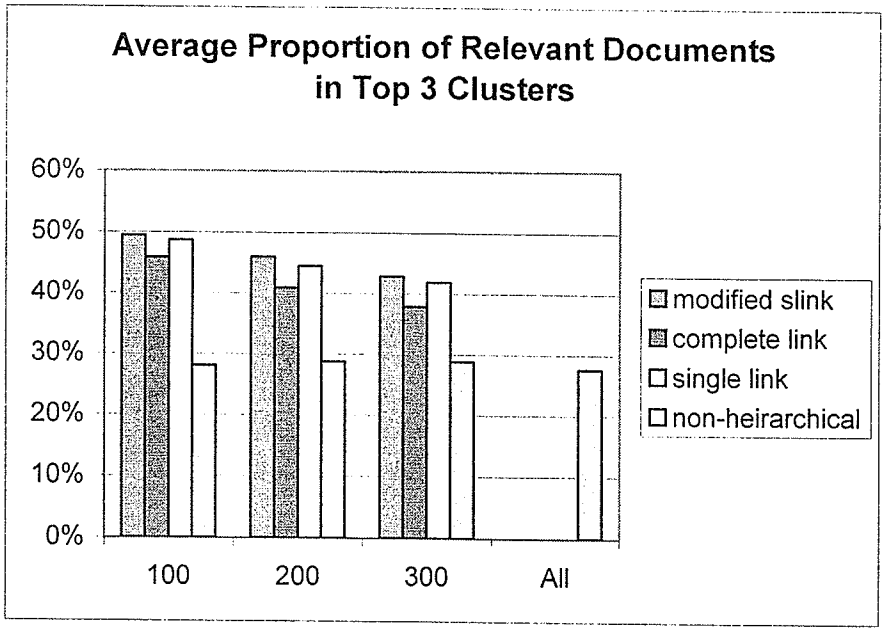
may not actually give us much information on the clusters themselves. This is why the clusters with the highest proportion of relevant documents need to be studied.

In IR, the two basic measurements of evaluation are precision and recall. Precision measures how relevant the retrieved documents are to a given query, while recall measure the percentage of all relevant documents that were retrieved. The measurement of the proportion of relevant documents within a cluster is similar to the measurement of precision. However this is only part of the picture and a similar measure of recall should also be used. If a cluster had just one document in it and it was relevant, it would have a relevant percentage of 100%. However, unless the original set of documents had just one relevant document in it, the cluster should not be considered a "good" cluster because it was unable to gather other relevant documents into it. One method that could be used to measure this is to study the number of relevant documents that actually end up in the clusters with the highest proportion of relevant documents.

The next section will look closer at the factors that affect the proportion of relevant documents found in a cluster. The section after that will study the factors that affect the number of relevant documents found in the clusters with the highest proportion of relevant documents.

3.4.1 Proportion of relevant documents

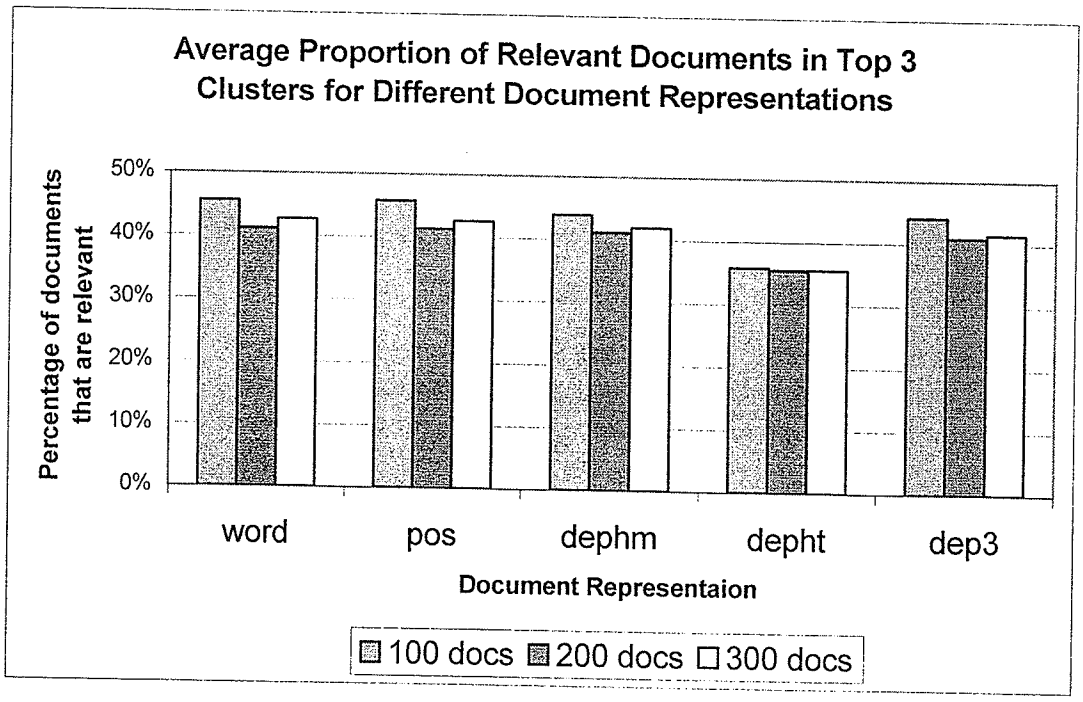
This section will look at the effectiveness of different clustering algorithms in terms of the proportion of relevant documents in its top cluster. The following chart shows the average proportion of documents that are relevant within the top three clusters formed by different clustering algorithms.



Just by looking at the above it would seem that the hierarchical clustering algorithms are outperforming the non-hierarchical clustering algorithm by quite a margin. However, we must remember that this chart is showing the cluster with the highest proportion of relevant documents. This means that if there was just one document in a cluster and that document was relevant, the cluster would be considered the cluster

with the highest proportion of relevant documents. Therefore, we cannot draw that conclusion simply from the chart above.

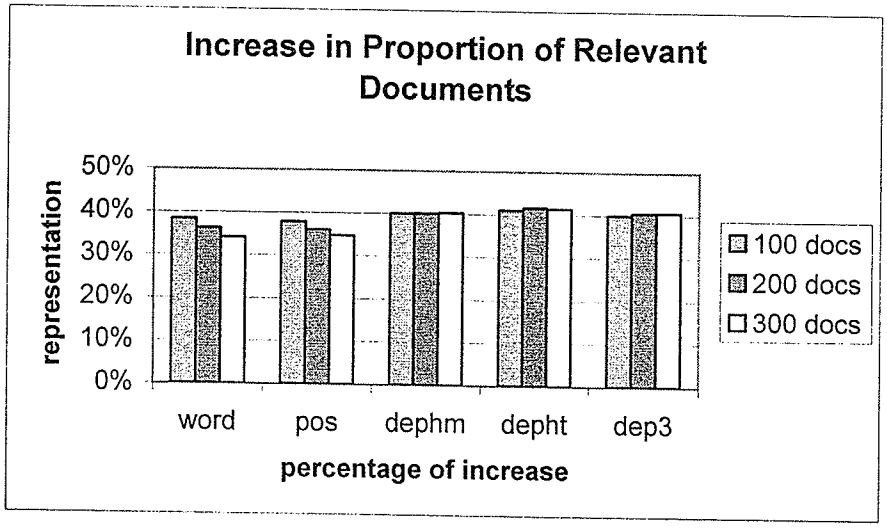
Earlier, it was found that a simple word representation had provided the best ordering for documents. Would a simple word representation produce the best clusters? The following chart illustrates the proportion of relevant documents in the top 3 clusters after clustering n documents. These values were calculated as the average of clustering n documents into different number of clusters for different representations and clustering algorithms.



Note that because only non-hierarchical clustering was used to cluster all documents, the result from that was not displayed in the previous chart as it would have a much lower average and be misleading. From the above we see that the results for the

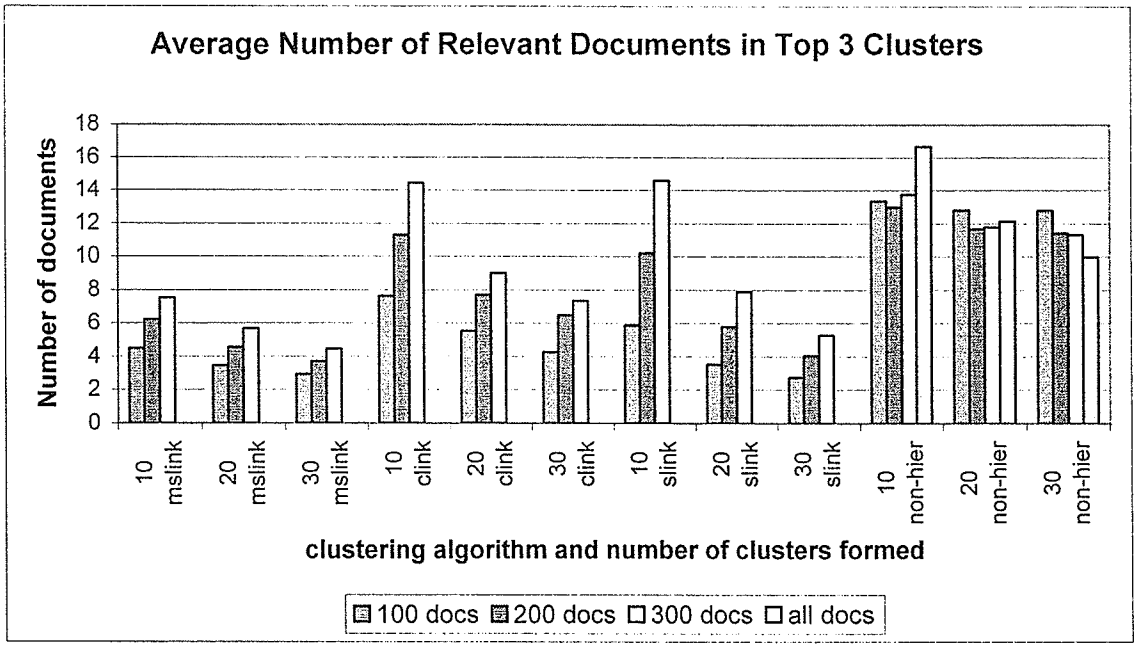
word, pos, dephm and dep3 representations are fairly similar. The depht representation seems to do worse than the others. Again these results are preliminary and until the actual number of relevant documents within a cluster are considered, it is hard to say whether or not different document representations affect clustering results.

Furthermore, recall that the word and pos representations had produced a better initial ranking of documents than the dependency representations. The percentage of relevant documents within their initial set was always several percent higher than the dependency representations. Thus, it should be expected that their clusters would have a higher proportion of relevant documents as well. If we were to look at the increase in the proportion of relevant documents for the different representations we would find that the dependency representations actually have a higher increase than word or pos. The chart below illustrates this.



3.4.2 The Number of Relevant Documents

The other measurement for a cluster has to do with the actual number of relevant documents within a cluster. This value is similar to the recall measurement in that it measures the number of documents found with respect to the number that could be found. The following graph shows the average number of relevant documents in the top 3 cluster for different clustering algorithms.



The first thing we should notice has to do with the hybrid clustering algorithm, mslink. The average number of relevant documents for mslink is really small (usually below two per cluster). Recall however that the percentage of relevant documents for this algorithm was quite high. This is an indication that the cluster that was formed was probably not very good. Consider the case of clustering 100

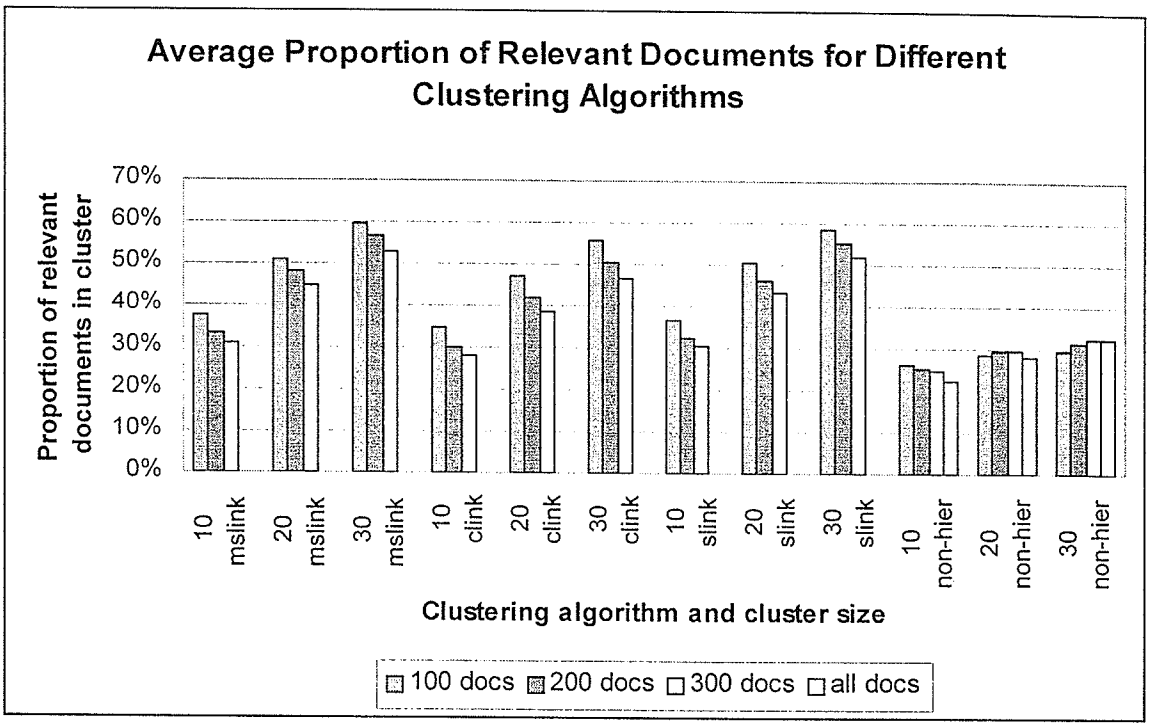
documents into 10 clusters. On average the proportion of relevant documents in the top three clusters is just a little below 50%. However, on average there was only around 4.2 documents within the top 3 clusters. If we take a look at the number of relevant documents in the set of 100 documents we would find that on average there would be around 20 documents. That is, less than 25% of all relevant documents were found within the top 3 clusters. Therefore, though the proportions of relevant documents within the top clusters are very high for mslink, the actual number of relevant documents found is fairly low. This indicates that the clusters formed for mslink were actually not very good. A similar argument also applies to the slink algorithm though to a lesser extent.

The last of the hierarchical clustering algorithms is complete link. In terms of proportion of relevant documents, the complete link algorithm had the lowest value of the hierarchical algorithms. However, in terms of the number of relevant documents, complete link was able to have better results than either single link or modified slink.

The clustering algorithm that was able to get the highest number of relevant documents in its top cluster was the non-hierarchical clustering algorithm. Thus it would seem that the non-hierarchical clustering algorithm produces a better recall value than non-hierarchical algorithms.

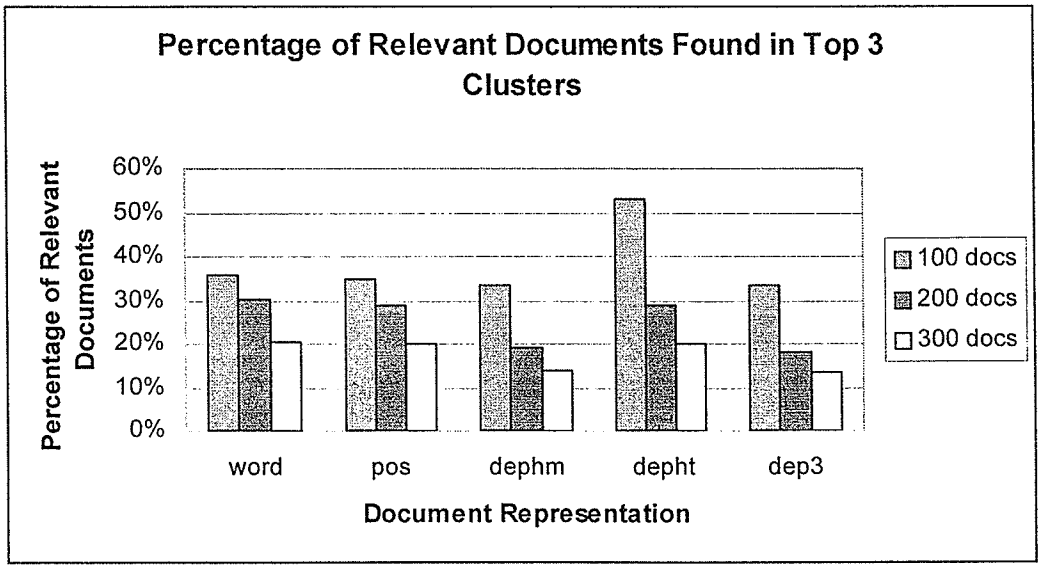
However, it should be noted that the non-hierarchical algorithm works in a different fashion than the hierarchical algorithms. The hierarchical algorithms begin with n clusters and group them until only m clusters are left. With the non-hierarchical clustering algorithm, no clusters are in place at the beginning. Each document is either placed into an existing cluster or begins a new cluster. What this means is that for non-hierarchical clustering, there are times when the number of clusters formed will be less than m because none of the documents seen were different enough from existing clusters to form that many clusters. However, this usually only happens when the m is 10% of n or less. Thus, if we were to look at the number of relevant documents for 200 docs and 10 clusters, the results should be fairly accurate.

When measuring precision recall values, precision generally decreases as recall increases. Although the non-hierarchical algorithm had produced clusters with a lower proportion of relevant documents (lower precision), it also produced clusters that had more relevant documents in it (higher recall). The question then is whether or not this is enough to explain the low proportion of relevant documents in the clusters. This is really hard to do because there is no set value for either of these measures within a cluster. However, one small estimate we could make is based on the fact that if we find two fairly similar recall values, we could then look at the corresponding precision values. In the above chart, the number of relevant documents in the top three clusters was fairly similar for clustering 300 documents into 10 clusters using clink, slink and non-hier. Consider the following:



The above chart is similar to the earlier chart on the proportion of relevant documents for different clustering algorithms. However, it is further broken down according to the number of clusters formed. From the above chart, we can observe that the precision for non-hierarchical algorithms is still slightly lower for a similar level of recall. In fact, the recall levels for the hierarchical algorithms were actually slightly higher. Thus, at least with the previous example, the non-hierarchical algorithm was still showing a lower performance than the hierarchical algorithms. However, this is just one value. Comparisons at different values of recall may yield different results. In any case, the above still shows that the non-hierarchical clustering algorithm does not perform at a significantly lower level than the hierarchical algorithms.

Earlier, the effect on the proportion of relevant documents for different document representations was considered. The number of relevant documents found within the top clusters should also be considered. However there is one difference that must be taken into account before this could be done. With the clustering algorithms, all clustering began with the same basic ordered list of documents. Thus, if 100 documents were clustered, the number of relevant documents within those documents didn't change for different clustering algorithms. However, for document representations this isn't the case. Different document representations will produce a different ordered list of documents to begin with. Thus the number of relevant documents in the initial list is also different. Therefore instead of considering the number of relevant documents present within the top 3 clusters, a better measure may be to consider the percentage of relevant documents that was actually placed into the top clusters. For example if initially there was 20 relevant documents in the list of 100 documents, what percentage of the 20 relevant documents was actually found within the top three clusters?



The above chart shows the percentage of all relevant documents that could be found in the top three clusters. When looking at different document representations, the precision for dephm and dep3 had been about the same as word or pos representations. When 100 documents were used the percentage of relevant documents in the top 3 clusters was fairly similar for these four clustering algorithms. However, as the number of documents clustered increases, the difference in this value between word/pos and dephm/dep3 is much more significant. Thus, the recall rate of the different clustering algorithms is most likely at least partly responsible for the high precision value for dephm and dep3.

However another possible contributing factor of why the precision for the dependency representations were the same may have to do with the size of the documents we are comparing. When doing the initial ordering, we compare each document with the query. Relative to the document, the query is usually very small. Dependency representations, and to a lesser extent part of speech representations, require a match on the entire term. Since the query is relatively small the probability of finding an exact match on a term is less likely than when using a word representation. When documents are clustered, they are being compared to other documents and not to the query. Thus, the probability of finding matching terms increases because documents are larger. It would be interesting to see if this was truly the case.

3.4.3 Conclusions

This work was attempting to study the clustering of documents. It looked at how different methods of representing the documents, similarity functions and clustering algorithms could affect the clusters formed. From the information gathered, several conclusions can be drawn.

The use of clustering to reorder search results would not lead to better precision recall values. The reason why using clustering to reorder the search results did not improve the ranking may have to do with how clusters are ordered after they are formed. Due to the fact that there is no way of knowing which cluster would have the more relevant documents, we could not reorder them based on anything other than its original similarity to the query. If a better method could be found to better predict how clusters should be ordered the use of clusters to reorder search results may still be improved.

Of all the similarity functions used, the info similarity function performed the best in ordering the documents. Its speed is comparable to that of dice or cosine providing that a simple method of estimating probabilities was available.

For the document representations studied, the simple word representation looked like it gave the best overall performance. The part of speech representation had a result very similar to that of word. However, the dependency representations typically did

worse than the word and part of speech representations. This evaluation of performance holds for both re-ordering of clustered documents and to the evaluation of the clusters themselves

Of the four clustering algorithms studied 3 were hierarchical and one was not. The three hierarchical algorithms produced clusters that had a higher proportion of relevant documents in them than the non-hierarchical algorithm. However these clusters tended to had fewer relevant documents in them. In other words, there was higher precision and lower recall for the hierarchical clustering algorithms. The hybrid algorithm that was tried did not produce very good clusters. It was expected that the results would fall between that of complete link and single link but this was not the case.

Though the non-hierarchical clustering algorithm did not produce clusters that were as precise as those produced by the hierarchical algorithms, it was able to produce clusters much faster than the hierarchical algorithms. The speed improvement is so great that depending on the situation, the reduction in precision may be worth the gain in speed. This is especially true if the number of documents to be clustered is large.

One criticism of non-hierarchical clustering algorithms is that the clusters they form depend on the ordering of the items to be clustered. [Willet, 1988] However, this may not be a bad thing because the documents are ordered according to its similarity

provide fewer variables in the resulting cluster and give a clearer picture of what is happening.

The methods of evaluating these clusters of documents are not very precise or standardized. Developing a more concrete method of cluster evaluation would be a great asset in determining the "goodness" of a cluster. Providing a standard method of evaluating clusters would allow better comparisons of results.

Bibliography

[Baker and McCallum, 1998] L. Douglas Baker and Andrew Kachites McCallum, *Distributional Clustering of Words of Text Classification*, to appear in SIGIR '98 Proceedings of the 21st annual international ACM SIGIR conference.

[Cutting et al., 1992], Douglass R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey, *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections*, SIGIR '92 Proceedings of the 15th annual international ACM SIGIR conference on Research and Development in Information Retrieval, pages 318-328

[Cutting et al. 1993], Douglass R. Cutting, David R. Karger, Jan O. Pedersen, *Constant Interaction-Time Scatter/Gather Browsing of Very Large Document Collections*, SIGIR '93 Proceedings of the 16th annual international ACM SIGIR conference on Research and Development in Information Retrieval, pages 126-134

[Frakes and Baeza-Yates, 1992] W.B. Frakes and R. Baeza-Yates, editors. *Information Retrieval, Data Structure and Algorithms*. Prentice Hall 1992

[Hearst and Pedersen, 1996] Hearst Marti and Jan O. Pedersen, *Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results*, SIGIR '96 Proceedings of the 19th annual international ACM SIGIR conference on Research and Development in Information Retrieval, pages 76-84.

[Hudson, 1990] Richard Hudson, *English Word Grammar*, Basil Blackwell, Oxford, UK, 1990

[Lin, 1993] Dekang Lin, *Principle-based Parsing without Overgeneration.* Proceedings of ACL-93, pages 112-120, Columbus, Ohio

[Lin, 1994] Dekang Lin, 1994. PRINCIPAR---An Efficient, broad-coverage, principle-based parser. In Proceedings of COLING-94. pp.42--488, Kyoto, Japan.

[Lin, 1998] Dekang Lin, *An Information Theoretic Definition of Similarity*, Proceedings of International Conference on Machine Learning, Madison, Wisconsin, July, 1998

[Lin] Dekang Lin, *MINIPAR Home Page*, Web Page, http://www.cs.umanitoba.ca/~lindek/minipar.htm

[Pirolli et. al, 1996] Peter Pirolli, Patricia Schank, Marti Hearst, and Christine Diehl, Scatter/Gather Browsing Communicates the Topic Structure of a Very Large Text Collection, Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, Vancouver, WA, May 1996

[Riloff, 1995] Ellen Riloff, *Little Words Can Make a Big Difference for Text Classification*, SIGIR '95 Proceedings of the 18th annual international ACM SIGIR conference on Research and Development in Information Retrieval, pages 130-136

[Salton, 1971] Gerard Salton, Editor, *The SMART Retrieval System Experiments in Automatic Document Processing*, Prentice-Hall, 1971

[Salton and McGill, 1983] Gerard Salton and Michael J. McGill, *Introduction to Modern Information Retrieval*, McGraw Hill 1983

[Silverstein and Pedersen, 1997] Craig Silverstein and Jan O. Pedersen, *Almost-Constant-Time Clustering of Arbitrary Corpus Subsets*, SIGIR '97 Proceedings of the 20th annual international ACM SIGIR conference on Research and Development in Information Retrieval, pages 60-66