Final Project: Synonym use in clustering

Kori MacCara

December 14, 1999

Contents

1.1 Introduction 2
1.2 Problem Description 2
1.3 Literature Review 2
1.4 A Possible Solution 5
1.5 Data Description 10
1.6 Setting the parameters 10
1.7 Testing Methodology 10
1.8 Testing Results 11
1.9 Discussion of Results 12
1.10 Relevance 13
1.11 Future Work 14
1.12 Conclusions 15

1.1 Introduction

The problem of document clustering has long been an issue in both the fields of machine learning (ML) and information retrieval (IR). [4]

From the ML perspective, clustering documents can be treated as a document classification problem. Each cluster or class in this case would have an associated set of properties (i.e. key-phrases which describe the content of the cluster). The task for such a problem would be to classify given documents into these classes. This is done most often in a supervised learning environment. [6]

From the IR perspective, clustering is usually done on the word level to identify words that often occur together within a document. Such knowledge can further aid the retrieval process by looking for words that often occur with the query terms. [4]

1.2 Problem Description

The clustering problem addressed by this paper is one from the machine learning domain, however, it uses techniques from the field of information retrieval. The goal is to group documents of similar content into groups or clusters in an unsupervised manner. Once clustered, the properties or each cluster can be utilized for many purposes. Some of these include:

- Document classification
- Generalization and or summarization
- Information retrieval

The goal of this project is to show that using word synonyms in a document's representation can improve the effectiveness of a clustering algorithm. Including synonyms in a documents representation involves finding synonyms of a given word such that these synonyms coincide with the context. For machines in an unsupervised environment, this is not a trivial task.

1.3 Literature Review

As mentioned before, the problem addressed by this paper is typically one that uses supervised learning techniques to achieve the desired clustering. One of the major drawbacks to this type of learning is that it requires large amounts of labeled data. [8] This labeled data is not only difficult to obtain or generate, but it is essential that the labels be non-contradictory (i.e. documents which fall under a given label are actually similar). It is possible that incorrect labels could result in the clusters not converging to a final set. [7]

It is for the above reasons that unsupervised clustering is desirable. Without having to label massive quantities of documents by hand, which is both tedious, time consuming, and possibly error prone, one could provide a clustering algorithm with documents and essentially walk away until it is done. This is a highly desirable scenario.

An algorithm that can accurately cluster documents (hopefully in a fashion similar to a clustering a human would generate) would have many practical applications. Assuming that the clustering can be achieved in a realistic amount of time, a clustering algorithm would be useful in many different applications, which include but are not limited to the following:

- Summary generation for documents of a given type
- Classification as interesting/uninteresting of new documents (based on experience (i.e. documents already classified)
- Generation of a document hierarchy or a hierarchical representation of a domain and the fields or areas it encompasses.
- Searching a domain for documents with subject matter on a given topic which can be represented in the domain subject hierarchy.
- Extended searches for documents online (using a crawler) for documents which fit into a given domain.

The problem of clustering documents in an unsupervised learning environment presents many difficulties. There are many measures and variable quantities that must be determined and set before any learning can begin. These are:

- Document representation
- Document similarity measures
- Clustering strategy
- Cluster representation
- Desired number of clusters
- Information to be stored by the clusters

Document Representation This is an important decision to be made in any system, whether it is based in the ML or IR fields. The internal document representation determines what can and cannot be done with a document within the system. Logically, a representation suitable to the purposes and goals of the system should be chosen. From the field of IR, there are many models which can be used to represent documents. The most commonly used are the boolean, vector, and probabilistic models.

- The boolean model consists of a set of words that occur in a document collection. Each document is represented by an array or ones and zeros indicating that the document contains the associated word or not, respectively.

- The vector model represents documents as list of words that occur in a document. Each word in a given document is given a weight that reflects it's importance level in the document and in document collection. An example of a weighting scheme would be TF.IDF.

- The probabilistic model represents a document as a series of terms which occur in the collection and their associated probabilities that the given term occurs in the document.

Document Similarity Measures These are methods which can be used to determine how closely one document resembles another. According to Manning and Shütze [4] there five common document similarity measures. These are (definitions assume a binary vector representation):

Similarity Measure	**Definition**
Matching Coefficient: Counts the number of terms that the documents have in common.	$X \cap Y$
Dice Coefficient: Normalizes the matching coefficient by the total of non-zero entries. The multiplier, 2, is used so that the result lies in the range [0.0 .. 1.0]	$\frac{2\lvert X \cap Y\rvert}{\lvert X\rvert + \lvert Y\rvert}$
Jaccard (or Tanimoto) Coefficient: Adds more of a penalty to the shared terms in the form of a proportion of all non-zero entries	$\frac{\lvert X \cap Y\rvert}{\lvert X \cup Y\rvert}$
Overlap Coefficient: Measures inclusion. This means that if one document is the subset of or the same as the other then they will be a perfect match.	$\frac{\lvert X \cap Y\rvert}{min(\lvert X\rvert, \lvert Y\rvert)}$
Cosine: Similar to the Dice coefficient, however it penalizes less when the number of non-zero entries is very different.	$\frac{\lvert X \cap Y\rvert}{\sqrt{\lvert X\rvert \lvert Y\rvert}}$

Clustering Strategy It is very important to devise a stable and robust clustering method. Since the determination of a grouping can seriously affect the effectiveness of a clustering algorithm. There are several known strategies that can be applied in an unsupervised learning environment.

- *Euclidean Distance* methods are based on explicitly defining the distance between two documents. It is the simplest form of clustering that can be done. [6, 9]

- *Probabilistic* methods which determine the probability of a document belonging to one of predetermined set of clusters [9] (these may be arbitrarily defined, depending on the initialization of the algorithm)
- *Entropy Minimization* methods attempt to minimize the overall entropy, or disorder, in a grouping of a document collection.[10] This means that these methods attempt to achieve an ideal number of clusters for the given data set, which maximizes the order or the collection.
- *Self-Organization Maps* try to create a non-linear projection of data onto a manageable grid.[3] Each point on the grid represents a possible point to which one or many documents can be mapped. Data is mapped to the grid using both statistical and probabilistic methods.

Cluster Representation is important as the representation chosen will have to be usable by both users and analysis tools. There are two types of cluster representations to choose from. These are:

- *Flat clusters* which are one layer clusters, to which a document either belongs or does not.
- *Hierarchical clusters* which can be multi-layered structures where a document can belong to a cluster, which in turn belongs to a larger cluster.

Desired Number of Clusters must be chosen depending on what type of clustering algorithm is chosen. However some, such as the Self-Organizing Map (SOM)[3] or the Entropy Minimization (EM)[10] algorithms, are able to converge to a number of clusters which best represents the collection.

Information to be Stored by the Clusters is important in cases where persistent records of clusters will be kept. This way the desired cluster information can be restored and used when called upon. Such indices can be useful in an information retrieval application when users are attempting to locate documents which deal with certain subject matter.

1.4 A Possible Solution

In order to simplify this problem it was assumed that all documents contained strictly text based information. This however reduces a document to words, all of the information contained in graphics, equations, diagrams, etc. is lost. It was also assumed that the directory structure of the document set can infer classification.

As this project created original software to preprocess and cluster the data set, it was necessary to start small. This meant choosing both a document representation and a clustering algorithm that are simple to minimize the amount of program level errors.

The first step in the process was choosing an acceptable document representation. The most common document representation in both IR and ML systems is known as the *bag of words* representation.[4] This is essentially the vector model (described in the previous section) from the field of IR.

Using the bag of words representation, a document, D, becomes a list of terms which are in D and their associated frequencies, tf[1], within D. For example a document could look like the following:

$$doc_j = < word_i : tf_{i,j} \ldots >$$

A technique used to reduce the amount of information stored is done through the removal of *stop-words*. These are commonly used words which do not contain any useful information about the document content (ex: and, or, the, followed, etc.). Another method from information retrieval called stemming was used to improve the document representation. Stemming essentially takes a word and returns the root or stem of the word. For example the stem of "computational" is "comput". This can also reduce the vocabulary of a system as various forms of the same stem do not need to be stored.

A common list of stop-words was obtained and used to filter documents so as to reduce the amount of information needed to be stored to described each document.[12] In order to perform word stemming on the documents the proven *Porter* [11] stemming algorithm was used. With each word in a document represented by a stem, the document in this system looks like the following:

$$doc_j = < stem_i : tf_{i,j} \ldots >$$

However this representation is not sufficient for the purposes of this project. It is important to ensure that the documents are normalized so that comparing a long document to another long document will not score better than the comparison of a long document to a short document that are equally dissimilar. It is also important to know how well a term within a document describes the content of the document. It is therefore useful to have some knowledge of how many documents contain the term, df[2], as well as how many times the term occurs in the document collection, cf[3].

To incorporate the document and collection frequencies into the document representation the *residual inverse document frequency* (RIDF) was used to scale the term frequencies. RIDF is defined as follows:

$$RIDF \;=\; IDF - \log_2(1 - p(0; \lambda_i)) \tag{1.1}$$

[1] *Term Frequency*, $tf_{i,j}$,is the number of occurrences of word,w_i, in document,d_j

[2] *Document Frequency*, df_i,is the number of documents in the collection in which a word, w_i, occurs.

[3] *Collection Frequency*, cf_i, is the total number of occurrences of a word, w_i, in the collection.

$$IDF = \log_2 \frac{N}{df} \quad (1.2)$$

$$p(k;\lambda_i) = e^{-\lambda_i}\frac{\lambda_i^k}{k!} \qquad \lambda_i > 0 \quad (1.3)$$

Equation 1.3 describes the *Poisson distribution* can be used to model the number of mistakes in a given document. According to Manning and Schütze [4] the most common model of the Poisson distribution is the average occurrence of a term per document, $\lambda_i = \frac{cf_i}{N}$. Here $p(k;\lambda_i)$ is used to to model the probability of a document containing exactly k occurrences of a given term. Therefore $N(1 - p(0;\lambda_i))$ is a reasonable estimator for df_i for non-content words, but overestimates df_i for content words.

The *inverse document frequency* (equation 1.2) gives full weight to terms that only occur in one document (as $\log_2 \frac{N}{1} = \log_2 N$). It is a good scaling factor to use when combining term frequencies and document frequencies of a given word into a single weight.[4]

Finally, equation 1.1 simply represents the combination of statistical and probabilistic weights. The residual inverse document frequency can be interpreted as the difference between the logs of the actual document frequency and the predicted document frequency. This residual can then be used as a term frequency scaling factor in the document representation. A document within the system then looks like:

$doc_j = < stem_i : tf_{i,j} * RIDF_i >$

Now armed with an acceptable document representation the task of clustering a set of documents can be tackled. Of the different algorithm types described in the previous section (Euclidean, probabilistic, EM, and SOM methods) an iterative Euclidean method was chosen for it's simplicity. This is hierarchical, which means the final result can be use as a generalization tree.

The algorithm used in this project is very similar to one described by Nilsson [9] (section 9.3.1). The algorithm used is described in table 1.1.

As an improvement on the above algorithm, the term synonyms were included in the document representation. This in itself introduced the problem of word sense disambiguation (WSD). For example, how can a machine know the difference between the following uses of the word "test":

1. I wrote a test today, I found it very difficult. I wonder if I passed.

   test: a set of questions or exercises evaluating skill or knowledge.

2. I went to the doctor today because my test results for my lungs came back

   test: trying something to find out about it

Cluster-Documents()

Iteratively calculates a predetermined number of clusters from the bottom up. Initially each document in the collection is considered to be contained within a cluster (i.e. $|C| = N$)

- Initialize the list of clusters, C, such that each document in the collection is a cluster ($|C| = N$)
- Initialize the desired number of clusters, N_c (user specified)
- While $N_c < |C|$. Do
  1. for each Cluster $C_i \in C$ do
     - for each Cluster $C_j \in C$ where $j \neq i$ calculate the Euclidean distance, $S_{i,j}$ between clusters C_i and C_j
  2. Choose the cluster pairs for which $S_{i,j}$ is minimized (i.e. C_i and C_j for $\min(S_{i,j})$).
  3. Create a new cluster, C_k out of C_i and C_j
  4. Remove clusters C_i and C_j from the list of remaining clusters
  5. Add the newly created cluster, C_k, to C
- Return final clustering C such that $|C| = N_c$

Table 1.1: Agglomerative clustering algorithm

Word disambiguation is important because each sense of the word test shown above has different synonyms associated with it which reflect it's meaning. The word communication is a synonym of the first sense of the word test whereas, experiment is a synonym of the latter.

Chandrasekar and Bangalore [1] describe a method to distinguish between words senses using a Maximum Entropy model. However for this project the simple method presented by Manning and Schütze [4] (Section 7.3.1) was favoured. This algorithm is an iterative process which safely assumes that the sense of a word is determined by the words surrounding it, it's context. The algorithm is presented in table 1.2.

Disambiguate (word w, context c)

> *An algorithm that can disambiguate between the senses of a word given a context using the definitions of the words in the context. It determines the sense of a word by making a comparison between the context word definitions and the definitions of each of the senses.*

- for all senses s_i in S of word w do
  1. for each word v_j in context c get it's definition, regardless of it's sense, E_{v_j}
  2. create the union of each definition, $\bigcup_{v_j \in c} E_{v_j}$. Remove stop-words. Stem each word in the union and remove the duplicates.
  3. find score for s_i where score(s_i) = overlap$(D_i, \bigcup_{v_j \in c} E_{v_j})$
- given sense, s, such that s = (arg max_{s_i} score(s_i)) find all synonyms, A, for s
- return A

Table 1.2: Lesk's dictionary-based disambiguation algorithm modified to return the synonyms of the chosen word sense

The various senses $s_1 \dots s_k$ of a word and their dictionary definitions $D_1 \dots D_k$ of an ambiguous word can be retrieved using WordNet. [5] WordNet is a lexical database of the English language created in the Cognitive Science Laboratory of Princeton University. WordNet is also capable of retrieving the synonyms of a given word if the sense is known.

1.5 Data Description

Since the task was to produce a unsupervised clustering algorithm, it was highly desirable to obtain a large document set that has been "pre-classified". This is because the actual classes of the documents must be compared to their assigned classes to determine the accuracy or effectiveness of the algorithm.

A dataset used by Nigam et.al. [8] was a good choice. This dataset consisted of approximately 1000 articles from each of 20 publicly available news groups. This provided a document base which was larger than necessary.

The news group dataset was, in a sense, pre-labeled as a class could be inferred from the directory structure. This is because each document was stored in a directory named for the news group to which it belonged.

1.6 Setting the parameters

Due to the choice of clustering algorithm only one parameter needed to be specified, the desired number of clusters. To further simplify the problem, a total of 2 clusters were produced. The only valid reason for choosing 2 as this number was for testing purposes, which are discussed in the next section.

1.7 Testing Methodology

Given the structure of the news group data set, it was possible to divide the set up into 380 ($\frac{20!}{18!}$) possible training sets. Each of the training sets consisted of two distinct news group article sets. To ensure complete independence of each test the training sets were restricted such that if a news group article set was used in a test, then it was considered unavailable for other test groupings. This reduced the possible testing sets to 190 ($\sum_{i=1}^{19} i$), which was larger than necessary.

From the news group dataset, 10 non-overlapping training sets were created. Each document filename was altered by prepending the news group name before being dumped into the test directory. This was done to facilitate verification once the algorithm assigned a cluster to the document.

In order to determine the value of adding synonyms to the document representation, the documents were first indexed without synonyms and clustered. This served as a baseline, or a starting point from which, any change in performance could be measured. The following observations were bench marked for each group:

- amount of CPU time taken to index files
- amount of CPU time taken to generate clusters

- number of correctly grouped documents for each class
- number of incorrectly grouped documents for each class

The time to index files and time taken to generate clusters are defined as the amount of time the CPU spent on the respective processes. Given two groups of documents which belong to two classes, C_i and Cj. A cluster is labeled as being of the class type for which it has the most elements, the remaining cluster is assumed to be of the other class. Therefore the number of correctly classified documents is the total number of documents whose determined class is the same as the pre-determined class type. Similarly, the number of incorrectly grouped documents is defined as the number of documents given a class type that differs from the pre-determined class of the document.

Once the benchmarking was complete, it was possible to perform a meaningful comparisons against the results on the data containing synonym information.

1.8 Testing Results

Each of the 20 news groups contained approximately 1000 documents. Due to time and resource constraints the size of a group had to be seriously reduced. To do this, 20 documents were randomly chosen from each group in order to get a representative selection. The new sets were then divided into 10 groups of 2 as follows:

1. alt.atheism and comp.os.ms-windows.misc
2. comp.graphics and comp.sys.ibm.pc.hardware
3. comp.sys.mac.hardware and misc.forsale
4. comp.windows.x and rec.autos
5. rec.motorcycles and rec.sport.baseball
6. talk.religion.misc and talk.politics.misc
7. sci.crypt and sci.med
8. rec.sport.hokey and sci.electronics
9. sci.space and talk.politics.mideast
10. soc.religion.christian and talk.politics.guns

Using the above groups of documents, the results observed during the benchmarking of the system are displayed in table 1.3. It should be noted that the times in this table are recorded in mm:ss.

Group	CPU time to index	CPU time to cluster	correct	incorrect
1.	4:03	6:40	11 and 12	9 and 8
2.	4:15	7:02	10 and 9	10 and 11
3.	4:07	6:54	11 and 11	9 and 9
4.	4:52	7:32	9 and 11	11 and 9
5.	4:34	6:43	12 and 10	8 and 10
6.	5:32	7:45	9 and 11	11 and 9
7.	4:48	6:57	11 and 10	9 and 10
8.	5:02	7:12	10 and 11	10 and 9
9.	5:26	7:12	12 and 11	8 and 9
10.	4:38	6:53	10 and 11	10 and 9

Table 1.3: Results of benchmarking

The results from the benchmarking test can be compared to the results observed during the project testing. The project testing results are located in table 1.4. It should be noted that the times recorded for the indexing in this table are of the format hh:mm:ss, where as the clustering time remains in mm:ss

Group	CPU time to index	CPU time to cluster	correct	incorrect
1.	10:43:28	12:01	12 and 8	8 and 12
2.	11:04:25	10:32	10 and 10	10 and 10
3.	10:32:36	10:21	9 and 11	11 and 9
4.	11:23:27	11:24	10 and 11	10 and 9
5.	11:17:28	12:20	10 and 10	10 and 10
6.	11:08:32	11:29	11 and 9	9 and 11
7.	10:54:22	12:08	11 and 7	9 and 13
8.	11:32:54	11:11	8 and 12	12 and 8
9.	11:57:24	11:45	10 and 10	10 and 10
10.	10:23:43	12:01	9 and 10	11 and 10

Table 1.4: Results of benchmarking

1.9 Discussion of Results

The classification results obtained during the benchmarking were for the most part, very poor. The percentages of correctly classified documents are very close to 50%, which is not much better than random when dealing with a two class problem. Therefore from the perspective of the benchmarking tests, anything would be an improvement.

During the testing of the synonym finding algorithm, it was observed that finding a synonym cost approximately 6 seconds. The effects of finding synonyms for all content words, can be easily observed when the project tests are compared against the benchmarks. The time for indexing took approximately 120 times longer. This is a very significant increase in time taken.

Although the indices generated were roughly twice the size of those created during the benchmarking, the clustering did not take that much longer. In fact on average they took approximately 4 minutes and 26 seconds longer than the benchmarks to complete. This is not a very large increase in clustering time, however it is still significant as it indicates as it reflects the expected correlation between the index size and the clustering time.

With the inclusion of synonyms in the document representation, it was expected that the document clustering would improve as with more information. However this turned out not to be the case in these tests. As it can be seen from the above tables, the number of correctly classified documents was on the average, reduced.

It is taught that this reduction in accuracy was caused by the performance of the word sense disambiguator. According to Manning and Shütze [4] the algorithm used in this project achieved accuracies between 50% and 70% when applied to a sample of ambiguous words. Some of the ramifications of using an inaccurate disambiguator are:

- Inclusion of unrelated words in a document description
- Possible diminution of document dissimilarities
- Introduces a large source of error to an algorithm that is already not much better than random classification.
- Skewed clusters (i.e. one cluster contains the majority of the documents, while the other only contains a few)

Another possibility that can be contributing to such poor performance is the assumption that the initial directory structure of the documents is accurate. This assumption implies that there is only one right way to cluster the given documents, when in fact there may be several acceptable cluster structures.

1.10 Relevance

Although this system fell short of the expected performance, there are many things that can be learned. The first of which is the necessity for a sound foundation upon which to build. The initial shortcoming of the clustering algorithm can be a result of many factors, including:

Document Representation The bag of words is not the only acceptable representation for a document, others could be used. Also there is the question of what weighting scheme to use for terms in a document.

Document Similarity measure As was shown earlier in this paper, there are many types of similarity measures between documents. It is therefore possible that the cosine similarity measure was not the best one to choose for this task.

The Algorithm for Clustering It is very likely that the Euclidean clustering algorithm used in this project did not find the *optimal* clustering of a document set. There are other, more complicated, algorithms that would likely perform much better.

As was mentioned earlier, the accuracy of the dictionary-based word disambiguator leaves much to be desired. Although its performance was poor, it did have some effect on the clustering algorithm. While these effects were considered to be negative, it can be theorized that improper choice of synonyms can adversely affect clustering. If this is correct then the inverse may also be true. This means that correctly choosing synonyms can improve clustering.

1.11 Future Work

This clustering system has left quite a lot of room for improvement. One modification that would undoubtedly improve the system as a whole would be the use of an EM algorithm to find the optimal cluster organization for a document set.

An EM-type algorithm can also be applied to the word sense disambiguation problem. According to Manning and Schütze, this type of disambiguator can reach accuracies of up to 95%. This is a significant improvement over the dictionary based method used in this project. Implementing such an algorithm in place of the current disambiguator would certainly be beneficial to the clustering mechanism.

Another approach to improving the word sense disambiguation is to use part of sentence (POS) tagging.[1, 2] A POS tagger could be used to preprocess the documents to be clustered. This would help to narrow the disambiguator's search for the correct sense because the word of interest will have been labeled as a noun, adjective, verb, or adverb.

Although another look at document representation may not be warranted, trying different representations or combinations there-of may prove to be more useful in the document clustering problem space. This can also be applied to the document similarity measures. Experimenting with different similarity measures will either lead to a better measure or prove the current measure is best.

It should also be noted that further examination of the data directory scheme will be done to ensure that the current clusters are accurate.

1.12 Conclusions

Although this project did not conclusively show that the use of synonyms in a document representation improves clustering, it did provide an indication that improperly chosen synonyms could adversely affect the clustering process. This in itself shows that synonyms can have an effect on the logical groupings of a document set. Furthermore, if synonyms for incorrectly chosen word senses impede clustering, then perhaps it is not to large of a leap to believe that correctly chosen synonyms could improve the structure of the clusters.

Bibliography

[1] Chandrasekar, R., & Bangalore, S. (1998) Knowing a word by the company it keeps: Using Local Information in a Maximum Entropy Model for Word Sense Disambiguation.
http://www.cis.upenn.edu/ mickeyc/cv/cv.html

[2] Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C. & Nevill-Manning, C.G. (1999) Domain-Specific Keyphrase Extraction.
http://www.cs.waikato.ac.nz/ gwp/publications/1999-Frank-et-al-IJCAI.ps.gz

[3] Kohonen, T. (1998) Self-Organization of Very Large Document Collections: State of the Art. *Proceedings of ICANN98, the 8th International Conference on Artificial Neural Networks*, *1*, pp. 65-74.

[4] Manning, C.D., & Schütze, H.. (1999) *Foundations of Statistical Natural Language Processing.* Cambridge: The MIT Press.

[5] Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1998) *WordNet: An Electronic Lexical Database.* Cambridge: The MIT Press.

[6] Mitchell, T.M. (1997) *Machine Learning.* Boston: WCB/McGraw-Hill, 1997.

[7] Niedermayer, D. (1998) An Introduction to Bayesian Networks and their Contemporary Applications.
http://www.gpfn.sk.ca/ daryle/papers/bayesian_networks/bayes.html

[8] Nigam, K., McCallum, A.K., Thrun, S., & Mitchell, T. (1999) Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, pp. 1-34.

[9] Nilsson, N.J. (1996) *An Introduction to Machine Learning*
http://robotics.stanford.edu/people/nilsson/mlbook.html

[10] Palubinskas, G., Descombes, X., & Kruggel, F. (1998) An Unsupervised Clustering Method using the Entropy Minimization. *Proceedings. Fourteenth International Conference on Pattern Recognition*, *2*, pp. 1816-1818.

[11] Porter, M.F. (1980) An algorithm for suffix stripping. *Program*, *14*(3), pp. 130-137.

[12] van Rijsbergen, C.J. (1979) *Information Retrieval.* London: Butterworths.