M.Sc. Thesis

Language Models for Distributed Information Retrieval

Christopher Graham McGiffen B.Sc. Hons

2000

A report submitted as part of the requirements for the degree of M.Sc. in Computing: Information Engineering at The Robert Gordon University, Aberdeen, UK.

Declaration

This is a declaration that:

1. This work has been composed by myself;
2. This work has not been accepted in any previous applications for a degree;
3. All verbatim extracts are distinguished by quotation marks;
4. All sources of information are specifically acknowledged.

Signed

Christopher G. McGiffen

Abstract

Distributed systems provide an efficient means for accessing large, dynamic information sources such as the Internet. Their modular nature comes from the ability to group related documents together and focus processing on the groups that are most relevant to a user's information need.

This report compares the use of two language modelling techniques for clustering documents. The first technique is based upon comparing document statistics to find similarities between documents and clusters, the second uses compression. It is shown that the second technique performs badly due to the poor identification of the key features that discriminate between clusters. However, several improvements are suggested that could make it more competitive.

Acknowledgements

I would like to thank Dr. W. J. Teahan and Prof. D. J. Harper for their guidance and support in the development of this report.

Contents

1 Introduction

The desire to distribute the various aspects of an information system comes from the need for accurate and speedy retrieval from vast information sources that are growing exponentially. Being able to group related documents is an important aspect in distributing such a system. Such groupings allow processing to be focused on certain areas of the system that are most relevant to the users' information need. The process of grouping these documents is known as clustering, and identifying the relevant groups known as collection selection.

This report looks at how language models can be used to create and represent clusters. Xu and Croft [18] have already used one form of language models that are based upon the statistical attributes of the modelled text. This report will compare this method with the use of a different language modelling technique that is based upon compression.

The objective of this research is to show that compression-based language models have the potential to be effective in distributed information retrieval. We will show that they can be successfully applied to clustering related documents, and that they are competitive with the technique that Xu and Croft [18] used.

This report will first look at related work that has been carried out using various modelling techniques. The focus will be on how histogram-based language models have been used in relation to distributed information retrieval, as well as look at the use of compression-based language models. In section 3, the experimental set up is discussed. This will give an overview of the document collections that are used, and consider the various issues involved in clustering. Section 4 looks at some of the preliminary investigations that were done with compression-based language models. This includes how individual documents are clustered and how well these models work at classifying documents. In section 5, results from the clustering experiments are reported. Finally, section 6 concludes with a summary of this report and reviews its achievements.

2 Related Work

This section will give an overview of the literature that is related to the use of language models in information retrieval. It will start with a brief overview of more traditional techniques that are used for modelling the information content in a retrieval system. It will then go on to give an overview of language models used in information retrieval and look in more detail at two particular language modelling techniques – histogram-based and compression-based. A more general overview of the traditional modelling techniques and language models can be seen in the full literature survey on distributed information retrieval, appendix B.

2.1 Traditional Modelling Techniques

There are various modelling techniques used to represent and search collections of documents. Three traditional approaches to modelling [4] are the *vector space* model, the *probabilistic* model and the *Boolean* model.

2.1.1 Vector Space Models

The vector space model [1] [4] is based upon the similarity between a document and a query. It represents documents and queries as vectors of terms and their frequencies, allowing individual terms to be weighted and manipulated. Each of the unique terms is represented as a dimension in a multi-dimensional space. It is the location of document and query vectors in this space that determines the similarity between them. Documents with similar semantic content as a query will appear closer to the query. Calculations based on geometry can therefore be used to measure the distance between the vectors and find the closest documents to a query. This method allows the partial matching of documents and queries.

2.1.2 Probabilistic Models

The probabilistic model [4] looks at the probability of a document being relevant to a query, using the distribution of terms over relevant and non-relevant documents. It works by assigning weights to each of the features in a document as a measure of how well that particular feature represents the document. The *tf.idf* function is often used for this measurement. It is the product of the frequency of the term in the document (*tf*) and the inverse frequency of documents that contain the term (*idf*).

2.1.3 Boolean Models

The Boolean model [4] is another example of an explanatory model. These models allow the use of Boolean algebra in formulating queries (such as the AND, OR and NOT operators). Traditionally the Boolean model could not rank returned documents. Extensions have however been proposed that can do this, as well as increase the ability of Boolean algebra for formulating the queries.

2.2 Language Models

Language models were originally used in the speech recognition community to model the statistical regularities in the generation of language. They have since then been successfully applied to a variety of domains, such as optical character recognition, statistical translation and spell checking. Recently these models have also been applied to information retrieval.

There are several benefits to using language models for information retrieval over other methods [10] [2]. Documents are modelled individually, and are not put into pre-defined classes for particular queries. There is no notion of relevance with language models. It is the probability of a document generating a query that is measured, not how relevant a document is to a query. A language model can also integrate the indexing and retrieval models into a single model. They are non-parametric, meaning they do not need any additional information to create the models. Instead they are entirely based upon the collection statistics.

Hiemstra and de Vries [4] relate the use of language models for information retrieval with the three traditional methods. They show that there is much in common between these methods, and that many of the algorithms used by the traditional techniques can also be applied to language models.

There are two forms of language modelling that are considered in this report – histogram-based language models and compression-based language models. The histogram-based language models, first exploited by Ponte and Croft [10], are based upon the individual words and how often they occur in a text. The next two sections will give an overview of both of these methods.

2.3 Histogram-based Language Models.

In information retrieval, histogram-based language models can be used to represent the probability of a word being used to describe a document. These models create a probability distribution $\{p_1, p_2 \ldots p_n\}$ over a vocabulary set $\{w_1, w_2 \ldots w_n\}$. The probabilities represent the likelihood of the term being generated by the document. To rank documents against a query, the probability of the document generating the query is calculated by combining the probabilities for query terms that are present in the document. Ponte and Croft [10] successfully apply this method to information retrieval, showing significant improvements over the *tf.idf* method. The probability distribution is created by calculating a maximum likelihood estimate $\hat{p}_{ml}$ for each of the terms *t* in document *d*:

$$\hat{p}_{ml}(t \mid M_d) = \frac{tf_{(t,d)}}{dl_d}.$$

For each of the terms *t* in a document model M_d, calculate the frequency of that term in the document $tf_{(t,d)}$ over the total document's length dl_d. The basic histogram-based language model proposed by Ponte and Croft does however have two problems [10]. The first is that when a query term does not appear in a document, the overall probability ends up being 0. This is known as the "zero frequency problem". To overcome this, the average probability of the term frequency cf_t over the entire collection size *cs* is taken into consideration when the term is not present in the document:

$$\frac{cf_t}{cs}.$$

The second problem is the confidence in the maximum likelihood estimate. Since documents can vary in size and content, a more robust estimate based on a larger amount of data is required. The mean probability of a term *t* in all the documents that contain it can be used:

$$\hat{p}_{avg}(t) = \frac{\left(\sum_{d_{(t \in d)}} p_{ml}(t \mid M_d)\right)}{df_t}.$$

This is calculated as the sum of all the probabilities of term *t* in all the documents that contain it, calculated from $p_{ml}(t|M_d)$, over the number of documents that contain it, df_t. This does however assume that all the documents are typical. To improve upon this, the frequency of a term in a document is compared to the normalised mean term frequency. In documents where a term frequency is typical of other documents, the mean term frequency is a safe measure. The further it moves from the mean, the riskier it becomes to use.

To smooth the estimate of the average probability for rare terms that do not appear in many documents, the estimate is based on all the terms that are equally as rare:

$$\hat{R}_{td} = \left(\frac{1.0}{(1.0 + \bar{f}_t}\right) \times \left(\frac{\bar{f}_t}{(1.0 + \bar{f}_t}\right)^{tf_{t,d}}$$

where $\bar{f}_t$ is the average frequency that the term *t* occurs in document *d*, and $tf_{t,d}$ is the frequency of term *t* in document *d*.

Song and Croft [13] also identify the need to ensure that a query term that is not found in the document does not result in a zero probability. They use an estimate that allocates some probability mass to such missing terms. Further expansion of the model is used to adjust the probability of missing terms based on information about the term in the rest of the collection. Another improvement made to the model takes into consideration the sequence of query terms, useful for checking for duplicates and also for phrases. A further improvement considered term pairing, where phrases of word pairs are used.

2.3.1 Cluster-based Language Models

Xu and Croft [18] define four methods for representing a distributed collection using language models. The documents are clustered to create *topics*. These topics are then represented using language modelling. Xu and Croft refer to these representations as *topic models*.

The *2-pass K means algorithm* is used for clustering related documents into groups. In the first pass, a portion of the documents is taken as the initial clusters. A distance metric is then used on each of the remaining documents to find the cluster that it is closest to. The second pass takes the results of the first pass as the initial clusters, and

goes through each of the documents to ensure that they are in their closest cluster. If not, then they are moved to the appropriate cluster.

The initial clusters can be selected in a variety of ways. Xu and Croft simply used the first few documents. Other possibilities involve randomly selecting documents, comparing documents to find the most similar/dissimilar or using hierarchical divisions/fusions of the collection until an appropriate number of clusters are created.

For measuring the difference between a document and a cluster, Xu and Croft use a modified version of the Kullback-Leibler divergence:

$$KL(d,c) = \sum_{f(d,w_i)\neq 0} \frac{f(d,w_i)}{|d|} \log \frac{f(d,w_i)/|d|}{(f(c,w_i)+f(d,w_i))/(|c|+|d|)}$$

where $f(d,w_i)$ is the frequency of word w_i in document d, $|d|$ is the size of document d, $f(c,w_i)$ is the frequency of word w_i in collection c and $|c|$ is the size of the collection.

Xu and Croft suggest four methods for organising the topics and topic models. The first method is *baseline distributed retrieval*, as shown in figure 2.1(a). Documents are not clustered into topics in this method; instead a topic model is created for each collection. This organisation represents how many existing distributed information retrieval systems currently organise their system. Collections are completely autonomous, and do not have to be changed in any way. However they are also heterogeneous, meaning that documents that will satisfy the user's information need could be spread over several sites.

The *global clustering* method, in figure 2.1(b), requires that all the collections are stored at a single site. They can then be combined and clustered to create tightly bound topics. These are then modelled to create the topic models. A single topic is likely to contain all of the documents that will satisfy a user's information need. However, the collections are centralised and there is no local autonomy over the individual parts.

Local clustering, shown in figure 2.1(c), clusters the documents in a collection to create topics, and then models these to create topic models. The collections have partial autonomy over the documents, as they will be stored at the local site. However the clustering does require control over how the documents are grouped. The topics

within a collection are therefore tightly bound, but topics that could satisfy the user's information need may be spread over several collections.

The final method, *multiple-topic representation*, is shown in figure 2.1(d). Like local clustering, each topic model represents a topic within a collection. However the collections have complete autonomy over documents, as they are not physically clustered. This does mean that the topic models can indicate the presence of a topic within a collection but will not have knowledge of where the particular documents that make up that topic are located. Documents that may satisfy the users need could also be spread over several collections.

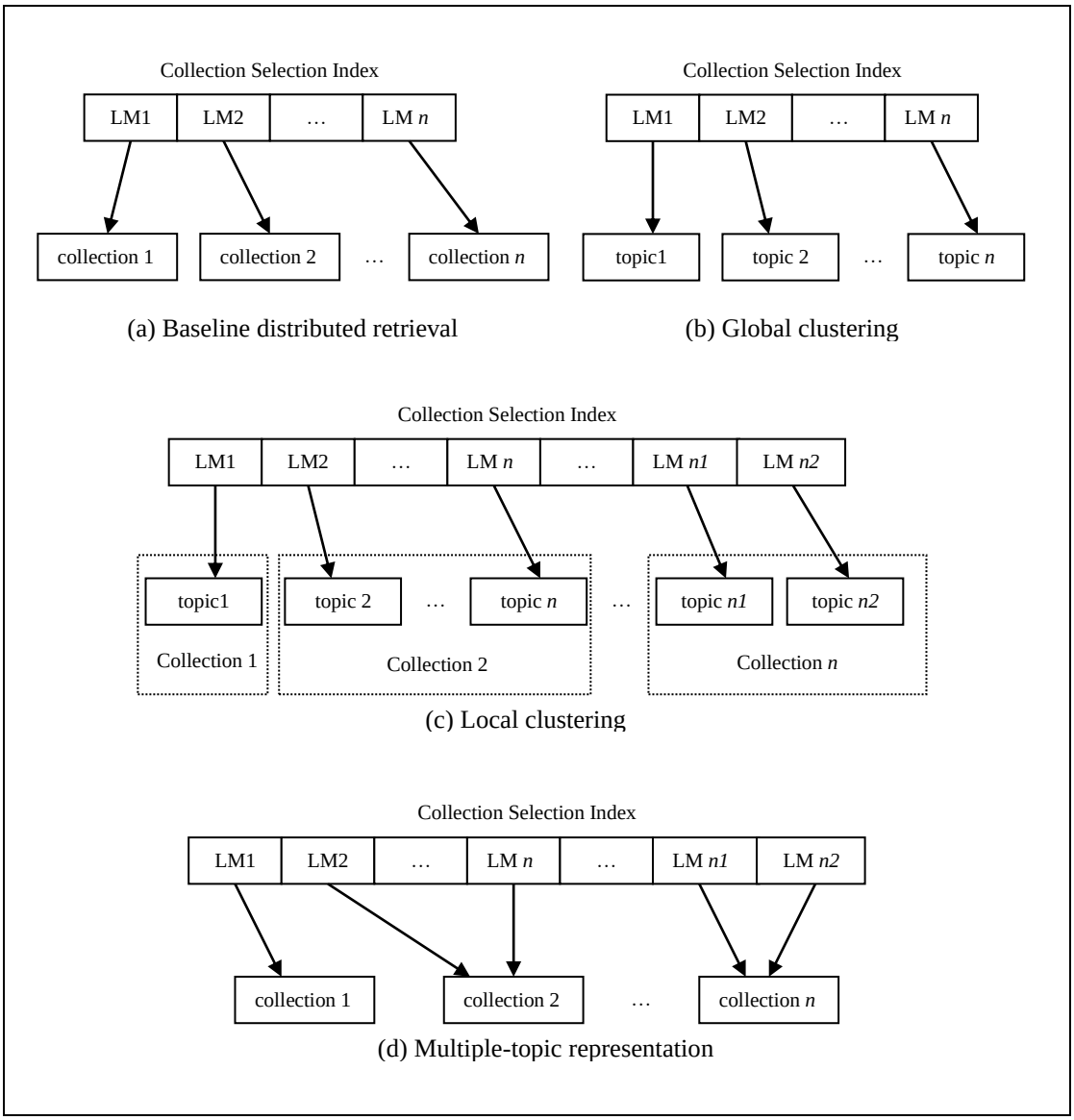

Figure 2.1 Four methods for organising cluster-based language models [18].

Xu and Croft compare these methods with centralised retrieval [18]. Each method ranks collections for several queries. They then search the top ten collections, retrieve 30 documents from each, and merge the results based on the document scores. The precision of the retrieved documents is then measured for various levels of recall. It was found that all the methods were improvements over the baseline distributed retrieval method. The global and local clustering methods were found to be very competitive compared with the centralised retrieval method.

2.4 Compression-Based Language Models

Compression-based language models look at the sequence of features in a text to predict upcoming features. These features can either be individual words and characters or entire sequences of words and characters. These models have been used in several applications, one of the most successful being text compression. They have also found some success in various natural language processing applications, such as language identification [3], spell checking and optical character recognition [6].

In this section, we will look at how compression based language models can be used to measure the difference between two documents. There is then an overview of the Prediction by Partial Match (PPM) compression scheme that has been successfully applied to language modelling [14]. This section will then finish with a look at some of the applications that compression-based language models have been applied to, and how their success relates to the potential use of compression-based language models in distributed information retrieval.

2.4.1 Minimum Cross-Entropy

The fundamental coding theorem [12] states that the average number of bits per symbol to encode a piece of text is given by its *entropy* [16]:

$$H(P) = -\sum_{i=1}^{k} p(x_i) \log\ p(x_i)$$

where the probabilities sum to one and there are k possible symbols with probability distribution $P = p(x_1), p(x_2),\ldots, p(x_k)$. When this is applied to the more general case of modelling a language with probability distribution L, it is known as the *entropy of a*

language, and can be considered as the limit to the entropy as the length of the message gets very large:

$$H(L) = \lim_{m\to\infty} -\frac{1}{m}\sum p(x_1, x_2, ..., x_m)\log\ p(x_1, x_2, ..., x_m).$$

It is unlikely that the true probability distribution of L will be known. However a model M can be used as an approximation to language L and obtain the upper bound to $H(L)$:

$$H(L, M) = -\sum p_M(x_1, x_2, \ldots, x_m)\log\ p_M(x_1, x_2, \ldots, x_m)$$

where the correct probability is $p(x_1,x_2,\ldots,x_m)$ and the probabilities are estimated by the model $p_M(x_1,x_2,\ldots,x_m)$. $H(L,M)$ is known as the *cross-entropy*, and can be used as a measure of how accurately the model is at representing the language. An upper bound to the entropy can be estimated by applying text compression:

$$H(L, M) = \lim_{m\to\infty} \frac{1}{m} b_M(x_1, x_2, ..., x_m)$$

where $b_M(x_1,x_2,\ldots,x_m)$ is the number of bits required to encode the string $x_1,x_2,\ldots,x_m$ using model M. This gives the number of bits per symbol required to encode a string of text from language L. Comparison of models can be achieved by calculating the cross-entropy for each model and the model with the smallest cross-entropy can be considered as the "best" model. The next section discusses the PPM compression scheme, one possible method for calculating the cross-entropy.

2.4.2 Prediction by Partial Match

PPM models use several fixed-order *context models* to predict upcoming symbols from those that have already been seen. These context models are known as "finite-context" models of order k, where k is the number of preceding symbols used to predict the upcoming one. A recommendation made by Teahan [16], based upon several studies with English and other common forms of text, concludes that an order 5 model performs competitively and that anything above order five will decrease performance. On the other hand, Frank, Chui and Witten [2] have found that using a maximum context length of order 2 for the specific problem of text classification works best.

Each of the context models used in a PPM model represent a particular order of *k*, ranging from the maximum context length to a default length of "-1", the base model that allocates a default probability to symbols based on the alphabet size. Table 2.1 gives an example of the state of a PPM model with a maximum context length of order 2 after the phrase "to be or not to be" has been processed.

Order $k = 2$			Order $k = 1$			Order $k = 0$			Order $k = -1$		
Predictions	c	p	Predictions	c	p	Predictions	c	p	Predictions	c	p
be → ␣	1	$^1/_2$	b → e	2	$^2/_3$	→ b	2	$^2/_{25}$	→ *A*	1	$^1/_{\|A\|}$
→ *Esc*	1	$^1/_2$	→ *Esc*	1	$^1/_3$	→ e	2	$^2/_{25}$			
						→ n	1	$^1/_{25}$			
e␣ → o	1	$^1/_2$	e → ␣	1	$^1/_2$	→ o	4	$^4/_{25}$			
→ *Esc*	1	$^1/_2$	→ *Esc*	1	$^1/_2$	→ r	1	$^1/_{25}$			
						→ t	3	$^3/_{25}$			
no → t	1	$^1/_2$	n → o	1	$^1/_2$	→ ␣	5	$^5/_{25}$			
→ *Esc*	1	$^1/_2$	→ *Esc*	1	$^1/_2$	→ *Esc*	7	$^7/_{25}$			
or → ␣	1	$^1/_2$	o → t	1	$^1/_7$						
→ *Esc*	1	$^1/_2$	→ r	1	$^1/_7$						
			→ ␣	2	$^2/_7$						
ot → ␣	1	$^1/_2$	→ *Esc*	3	$^3/_7$						
→ *Esc*	1	$^1/_2$									
			r → ␣	1	$^1/_2$						
o␣ → b	2	$^2/_3$	→ *Esc*	1	$^1/_2$						
→ *Esc*	1	$^1/_3$									
			t → o	2	$^2/_5$						
r␣ → n	1	$^1/_2$	→ ␣	1	$^1/_5$						
→ *Esc*	1	$^1/_2$	→ *Esc*	2	$^3/_5$						
to → ␣	2	$^2/_3$	␣ → b	2	$^2/_9$						
→ *Esc*	1	$^1/_3$	→ n	1	$^1/_9$						
			→ o	1	$^1/_9$						
t␣ → t	1	$^1/_2$	→ t	1	$^1/_9$						
→ *Esc*	1	$^1/_2$	→ *Esc*	4	$^4/_9$						
␣b → e	2	$^2/_3$									
→ *Esc*	1	$^1/_3$									
␣n → o	1	$^1/_2$									
→ *Esc*	1	$^1/_2$									
␣o → r	1	$^1/_2$									
→ *Esc*	1	$^1/_2$									
␣t → o	1	$^1/_2$									
→ *Esc*	1	$^1/_2$									

Table 2.1 PPMC model for "to be or not to be"
(Character based, maximum order 2)

Each context model records the symbols that have followed every sequence of length *k* that have so far been seen. They also keep count *c* of how often these have been

seen, and calculate the prediction probability p from these. For example, in the context model with $k = 2$, the various possible context sequences of two symbols, such as “be”, are shown along with the possible symbols that have so far been seen to follow them; in this case the only other symbol to follow “be” is a space (␣) symbol.

The probabilities from each of the symbols in a sequence are used to predict the upcoming characters. Symbols are encoded from a combination of the probability distributions from each context model, known as *blending*. This blending is achieved by *escape probabilities*. Encoding starts with the largest order context model. If the symbol has never occurred in this context, the escape probability is used and the process moves to the context model at the next order down. This continues down the orders until the symbol is found, at which point it can be encoded relative to the probability distribution predicted by this context. Formally, given $S = c_1, c_2, \ldots, c_m$, then the probability of sequence S is given by:

$$p(S) = \prod_{i=1}^{m} p'(c_i \mid c_{i-2}, c_{i-1})$$

where p' gives the probabilities returned by the order 2 PPM character model.

character	probabilities encoding (without exclusion)	probabilities encoding (with exclusion)	codespace occupied
␣	$^1/_2$	$^1/_2$	$-\log_2 {}^1/_2 = 1$ bit
t	$^1/_2$, $^1/_2$, $^3/_{25}$	$^1/_2$, $^1/_1$, $^3/_{20}$	$-\log_2 (^1/_2 \cdot {}^1/_1 \cdot {}^3/_{20}) = 3.7$ bits
a	$^1/_2$, $^1/_2$, $^7/_{25}$, $^1/_{\lvert A\rvert}$	$^1/_2$, $^1/_1$, $^6/_{20}$, $^1/_{\lvert A\rvert-7}$	$-\log_2 (^1/_2 \cdot {}^1/_1 \cdot {}^6/_{20} \cdot {}^1/_{249}) = 10.7$ bit

Table 2.2 Example encoding using the PPM model described in Table 2.1.

The probabilities associated with the escape sequence can be calculated in a variety of ways. The method used in the example is called “Method C”, and is based upon the number of unique symbols seen in the context. Table 2.2 gives three examples of encoding a symbol using the PPMC model described in Table 2.1. For example, consider encoding the *t* symbol. The order 2 context model is first used for the context “be”. However, there has never been a *t* to follow this, so the escape probability is taken, in this case $^1/_2$. At the next order down, the context “e” is used but again there has never yet been a *t* symbol to follow this, and the escape probability is taken again. At the order 0 model, the *t* symbol is found and the probability $^3/_{25}$ is taken. These probabilities are then combined to give an average of 3.7 bits required to encode each symbol.

Table 2.1 also shows the use of *exclusion*, used to rule out all the impossible sequences. An example of exclusion is shown when encoding the symbol *t*. After escaping from the order 2 context model, it is known that the ␣ symbol no longer follows the *e*, as this has already been predicted by the order 2 model. As such the probabilities can be updated to exclude this possibility. Another method that improves prediction performance is *update exclusion* [9]. In this method the counts themselves are only updated if the higher order models have not already predicted the sequence.

Order $k = 1$				Order $k = 0$		
Predictions		c	p	Predictions	c	p
be	→ or	1	$^1/_2$	→ be	2	$^2/_{10}$
	→ *Esc*	1	$^1/_2$	→ not	1	$^1/_{10}$
				→ or	1	$^1/_{10}$
not	→ to	1	$^1/_2$	→ to	2	$^2/_{10}$
	→ *Esc*	1	$^1/_2$	→ *Esc*	4	$^4/_{10}$
or	→ not	1	$^1/_2$			
	→ *Esc*	1	$^1/_2$			
to	→ be	2	$^2/_3$			
	→ *Esc*	1	$^1/_3$			

Table 2.2 PPMC model for “to be or not to be”
(Word based, maximum order 1)

PPM models are not restricted to using individual characters as a representation of features in a text. They can also be applied to entire words. For example, table 2.3 shows a PPMC order 1 word model after processing the string “to be or not to be”. Each of the six words in the model are treated as a separate symbol, just as the nine characters in the character model are in table 2.1. Unlike the character model though, each context uses one or more words in a sequence, and go on to predict entire words rather than just a single character.

2.4.3 Applications of PPM

Several applications have been demonstrated for PPM. Teahan [16] shows that it can be successfully applied to language identification, authorship ascription and classification by genre. Frank, Chui and Witten [2] successfully use PPM for text categorisation, and show that it is competitive with state of the art text categorisation

techniques[1]. These applications all show that PPM can be successfully used to identify relationships between documents. This signifies that they have the potential for use in various aspects of distributed information retrieval, such as grouping together related documents into clusters and finding the best cluster for a particular query.

For language identification, Teahan uses the Book of Genesis from the Bible in six different languages. For each version, the first 10,000 words are put aside for testing, and the remaining words are used to train a model. These models are based upon an order 5 PPM character model. The cross-entropy is then calculated for each of the six test texts when compressed with each of the six models. Table 2.3 shows the results that are achieved. This table shows that the model that is trained on the same language as the text that is being compressed gives the minimum cross entropy. This therefore proves the idea that models trained on text from various languages can be used to identify the language of a piece of text. Teahan goes on to apply this technique to identifying the period of a text, (i.e. old, middle or early modern English), and dialect (i.e. British-English or American-English) with almost equal success.

Original Text	*Untrained*	*English*	*French*	*German*	*Italian*	*Latin*	*Spanish*
English	1.97	**1.56**	2.30	2.38	2.30	2.33	2.29
French	2.13	2.54	**1.71**	2.56	2.54	2.59	2.56
German	2.14	2.64	2.60	**1.75**	2.55	2.59	2.59
Italian	2.26	2.67	2.66	2.65	**1.83**	2.61	2.68
Latin	2.45	2.87	2.91	2.91	2.82	**1.97**	2.84
Spanish	2.19	2.57	2.61	2.61	2.57	2.57	**1.75**
Size of training text (chars.)		180,359	175,331	191,840	170,923	149,121	169,461

Table 2.3 Identifying the text for six different languages [16].

To show how PPM can be applied to authorship ascription, Teahan addresses the problem of identifying the authors of the Federalist Papers. This famous problem concerns a set of short essays written by three different authors under the same pseudonym during the 18th century. The authors of most the documents have been agreed upon; however, there are 12 that remain in flat dispute between two of the authors, Madison and Hamilton. Two models are trained using a portion of the documents that have been credited to each of the two authors, each model representing an author. Table 2.4 shows the results from compressing the disputed

[1] Although they stated that their results were negative, results achieved with the PPM models are within range of the state of the art techniques.

documents using the two models. These results show what many historians claim, that Madison was the main contributor of these documents. Historians are however uncertain about documents 62 and 63, and this is also reflected in the results from the near equal values of cross-entropy that come from each of the models. Some of the documents where the authorship is known are also compressed against the two models. Table 2.5 shows that these results are mostly as expected, with almost every document being correctly ascribed to the author who wrote it.

Document Number	*Madison (bpc)*	*Hamilton (bpc)*	*Document Number*	*Madison (bpc)*	*Hamilton (bpc)*
49	**1.79**	1.93	55	**1.86**	1.97
50	**1.92**	2.07	56	**1.70**	1.85
51	**1.72**	1.88	57	**1.85**	1.98
52	**1.78**	1.94	58	**1.80**	1.93
53	**1.79**	1.93	62	**1.83**	1.84
54	**1.73**	1.89	63	**1.82**	**1.82**

Table 2.4 Identifying the author of the disputed documents [16].

Papers known to have been written by Madison			*Papers known to have been written by Hamilton*		
Document Number	*Madison (bpc)*	*Hamilton (bpc)*	*Document Number*	*Madison (bpc)*	*Hamilton (bpc)*
44	**1.76**	1.90	59	**1.82**	1.88
45	**1.67**	1.81	60	1.78	**1.71**
46	**1.78**	1.85	61	1.78	**1.73**
47	**1.70**	1.81	65	1.87	**1.82**
48	**1.85**	1.99	66	1.80	**1.75**

Table 2.5 Identifying the author of known documents [16].

Teahan also takes a preliminary look at how PPM can be used for classifying documents by genre. Twenty models are created to represent the 20 newsgroups that are in the Newsgroups data set [8]. The collection contains 20,000 documents, of which 80% are used to train the 20 models. The main bodies of the remaining 20% of the documents are then compressed against the 20 models. The correct category was chosen for 82.1% of the articles. Where articles are mis-classified, the correct model generally has a very small difference in cross-entropy to the chosen model. This is shown by the number of correct classifications when considering the top two models, which improves accuracy to 91.5%, and when considering the top three models, which improves accuracy further to 94.2%.

Frank, Chui and Witten [2] perform text categorisation using the Reuters-21578 [7] newswire stories. This collection is made up of 12,902 stories, averaging 200 words each. There are 118 pre-defined categories and each document has relevance

judgements on what categories it is relevant to. The number of documents per category is however highly skewed, with ten of them containing 75% of the entire collection.

The first experiment compares two categories against each other. Documents from the two categories are split into a training set and a testing set. Two models are created to represent each of the two categories, and trained on the documents from each training set. The test documents are then compressed against each of these models, and the difference between the compression ratios is calculated. Whether this value is positive or negative indicates if the correct model is chosen. This experiment is applied to the Reuters collection, and the results achieved by the ten largest categories are considered in detail.

	corn	corp. acq.	crude oil	earn-ings	grain	inter-est	mon. mrkt.	ship-ing	trade issues	wheat
corn	-	0.43	0.39	0.53	0.00	0.62	0.50	0.38	0.45	0.09
corp. acq.	0.79	-	0.40	0.31	0.59	0.65	0.63	0.47	0.61	0.68
crude oil	0.45	0.26	-	0.35	0.37	0.48	0.43	0.37	0.38	0.46
earnings	1.47	0.99	1.15	-	1.18	1.53	1.48	1.34	1.39	1.27
grain	0.13	0.36	0.33	0.47	-	0.54	0.44	0.33	0.41	0.11
interest	0.65	0.26	0.33	0.36	0.46	-	0.17	0.52	0.55	0.59
mon. mrkt.	0.57	0.39	0.37	0.50	0.44	0.18	-	0.46	0.32	0.55
shipping	0.27	0.19	0.16	0.32	0.20	0.38	0.29	-	0.31	0.24
trade issues	0.36	0.32	0.25	0.45	0.26	0.33	0.20	0.35	-	0.34
wheat	0.11	0.37	0.35	0.49	-0.06	0.58	0.46	0.33	0.45	-

Table 2.6 Mean differences in compression between two models [2].

Table 2.6 shows the mean difference in compression between the two categories when encoding testing documents using a model built from training documents relevant to the row category. Results are encouraging, with the majority of documents being correctly classified, indicated by a positive number in the table. The one point where the mean difference in compression is actually negative comes from encoding the *wheat* documents using the *wheat* and *grain* models. Similarly there is no difference between *corn* and *grain*, and very little difference between *corn* and *wheat*, *grain* and *corn*, *grain* and *wheat* and *wheat* and *corn*. These categories are closely related, and indicate that compression-based language models could have difficulty when it comes to classifying documents that are relevant to multiple categories.

A further experiment uses a positive and negative model to represent each category. These are trained on relevant and non-relevant documents, and test documents are compressed against both of these models for every category. It is the difference

between the two compression ratios that makes the decision on whether the document belongs in this category or not. This means that the decision is completely separate from whether the document belongs to other categories, allowing a document to be considered as belonging to several classes. Results are again encouraging, but still show much confusion where categories are closely related, such as *grain*, *wheat* and *corn*. The results from this experiment are also compared against two other state of the art categorisation techniques. These two methods are shown to perform better than PPM because they are much more sensitive to individual features in the text. Frank, Chui and Witten conclude that without better feature discrimination, compression based categorisation schemes will always perform poorly when compared to state of the art techniques with documents that rely on a few discriminating features.

3 Experimental Set-up

To show the efficacy of using compression-based language models in distributed information retrieval, several experiments were considered. Some of these look at validating the use of compression-based language models for information retrieval (section 4 looks at these in more detail). However, the main objective of this research is to compare the ability of compression-based language models with the histogram-based language models used by Xu and Croft [18]. As such, the main focus of this report will be on the experiments used to compare clustering performance using both histogram-based language models and compression-based language models.

This section will describe the two document collections that have been used for the various experiments. It will then look at the various issues involved in setting up the clustering experiments, with a look at how the documents were prepared for clustering, how the initial clusters were seeded from a selection of the documents in the collection, and how the clustering was performed.

3.1 The Data

Two document collections were used in the various experiments carried out in this report. They are the Reuters collection and a sub-collection of the TREC data set. Both offer a large set of documents covering several topics. They also come with judgements on which documents are relevant to which topics. This section gives a brief description of these two collections.

3.1.1 The Reuters Collection

The Reuters collection [7] contains 21,578 documents that were assembled and indexed by Reuters Ltd. It was made available to the general public to assist in the research and development of information retrieval systems. The collection uses SGML tagging to ensure a consistent formatting over all the documents. The documents have been sorted into chronological order and each given a unique identifier.

Figure 3.1 shows a typical document from the Reuters collection. Each document is represented as the text between the <REUTERS> and </REUTERS> tags, and is identified by the NEWID value. The topics that the document is relevant to are

identified between the <TOPICS> and </TOPICS> tags, with each topic surrounded by <D> and </D> tags. The main body of text is identified by the <BODY> and </BODY> tags.

```
<!DOCTYPE lewis SYSTEM "lewis.dtd">
<REUTERS
    TOPICS="YES"
    LEWISSPLIT="TRAIN"
    CGISPLIT="TRAINING-SET"
    OLDID= "08707"
    NEWID="08707">
<DATE> March 24 </DATE>
<TOPICS><D> earn </D></TOPICS>
<PLACES><D> London </D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN></UNKNOWN>
<TEXT>
<TITLE> WOOLWORTH U.K. SAYS GROWTH PROSPECTS EXCITING </TITLE>
<DATELINE> LONDON, March 24 </DATELINE>
<BODY>
Woolworth Holdings Plc WLUK.L which
earlier announced a 1986 pre-tax profits rise of 42 pct over
1985, said its prospects for growth were very exciting.
   The profit figure of 115.3 mln stg exceeded a forecast by
some 10 pct made during the hostile bid by Dixons Group Plc
DXNS.L last year and the company said the results were a
major step towards the aim of making Woolworth the most
profitable retailing group in the U.K.
   It ...
</BODY>
</TEXT>
</REUTERS>
```

Figure 3.1 Example document from the Reuters collection.

The collection that is used in these experiments is a subset of the entire collection based on the standard Mod Apte split. This sub-collection is made up of 9603 documents from 115 different categories. The majority of these documents are judged relevant to a single category, however there are some that are relevant to several categories and some that have no relevance judgement at all.

3.1.2 The TREC Collection

The TREC collection [17] has been created to assist evaluation of text retrieval systems. The collection is made up of several sources spanning several years (table 3.2), taking up almost 6Gbytes of computer storage. It consists of documents, topics and relevance judgements of which documents are relevant to which topics.

The documents are tagged using SGML for simple parsing. Although they all follow the same major structure, minor differences do exist in the tagging used between the sources. It is important to ensure that documents follow the same format so that they can be easily processed. Pre-processing done by TREC ensures that the basic format is followed, but does not go so far as correcting grammatical and minor formatting errors. This is to mimic some of the difficulty faced in real world information retrieval where documents will not always necessarily be error free.

	Size (MB)	# Docs	# Words / Doc
Disk 1			
Wall Street Journal, 1987-89	267	98,732	434.0
Associated Press newswire, 1989	254	84,678	473.9
Computer Selects Articles, Ziff-Davis	242	75,180	473.0
Federal Register, 1989	260	25,960	1315.9
Abstracts of US DOE publications	184	226,087	120.4
Disk 2			
Wall Street Journal, 1990-92	242	74,520	508.4
Associated Press newswire, 1988	237	79,919	468.7
Computer Selects articles, Ziff-Davis	175	56,920	451.9
Federal Register, 1988	209	19,860	1378.1
Disk 3			
San Jose Mercury News, 1991	287	90,257	453.0
Associated Press newswire, 1990	237	78,321	478.4
Computer Selects articles, Ziff-Davis	345	161,021	295.4
US Patents, 1993	243	6,711	5391.0
Disk 4			
The Financial Times, 1991-94	564	210,158	412.7
Federal Register, 1994	395	55,630	644.7
Congressional Record, 1993	235	27,922	1373.5
Disk 5			
Foreign Broadcast Information Service	470	130,471	543.6
The LA Times	475	131,896	526.5
Routing Test Data – Disk 6			
Foreign Broadcast Information Service	490	120,653	581.3

Table 3.2 The TREC Collection [17].

The topics provide “user need” statements rather than more traditional queries. This is to allow a wider variety of query methods to be used built from the statements, as well as increase the amount of information that can be made available to each topic. The topics in TREC6 are made up of the topic number, a title, a description of the topic and a narrative stating what documents must contain to be considered relevant to the topic.

Relevance judgements are necessary to allow the evaluation of systems that use the collection. Each of the participating retrieval systems at the TREC conference submits the results from runs that are based on the given topics. A pool of documents from

these results is created and then shown to human assessors. The relevance judgements are represented as a list of the pooled document identities and the topics they are relevant to.

Rather than using the entire TREC 6 collection, a single part of that collection was used in the experiments. This choice was based on a variety of factors. These included ensuring the chosen collection was not so large that it took too long to process, and yet ensuring that it contained enough documents to create several large clusters. The average size of documents was also an important factor. Larger documents will most likely contain more discriminating features to assist distinguishing documents from each other. The collection chosen is the Federal Register source from 1988 on disk 2 (table 3.2). This is one of the smaller collections, 209 Mbytes in size, and it does also contain relatively large documents, an average of 1378.1 words per document. There are also 19,860 documents in this collection, enough to create several large clusters.

```
<DOC>
<DOCNO> FR88108-0001 </DOCNO>
<DOCID>fr.1-08-88.f2.A1000</DOCID>
<TEXT>
<FTAG tagnum=4700></FTAG>
<ITAG tagnum=90>
<T4>Federal Register</T4> / Vol. 53, No. 5 / Friday, January 8, 1988 /
Rules and Regulations
<ITAG tagnum=1>Vol. 53, No. 5</ITAG>
<ITAG tagnum=2>Friday, January 8, 1988</ITAG>
<ITAG tagnum=50>DEPARTMENT OF AGRICULTURE</ITAG>
<ITAG tagnum=10>
<T2>SUPPLEMENTARY INFORMATION: </T2>This final rule is issued under Marketing
Order 907 (7 CFR Part 907), as amended, regulating the handling of navel
oranges grown in Arizona and designated part of California. This order
is effective under the Agricultural Marketing Agreement Act of 1937, as
amended, hereinafter referred to as the Act.
This ...
</TEXT>
</DOC>
```

Figure 3.2 Example of a document from the FR88 collection.

Figure 3.2 shows a typical document from the TREC FR88 collection. The document is represented as the text between the <DOC> and </DOC> tags. The text between the <DOCNO> and </DOCNO> tags is the documents identifier. The main body of text is contained in between the <TEXT> and </TEXT> tags. Most documents in this collection follow a similar structure of providing document details, such as its issue or department, then giving paragraphs on ACTION, SUMMARY, DATES, FURTHER INFORMATION CONTACT and SUPPLEMENTARY INFORMATION. These

paragraphs are identified by the <T2></T2> tags, with one of the headings in between these tags. TREC FR88 uses several other tags throughout the body of each document that provides additional formatting information.

3.2 Design Issues

There were several issues involved when designing the clustering experiments. Issues such as how documents should be prepared, how to select seed documents for initial clusters and how to actually perform the clustering all had to be considered when designing the experimental process. This section will go on to look at each of these issues.

3.3.1 Preparation of Documents

Documents had to be prepared before clustering. This involved extracting the text from both the Reuters files and the TREC files, as well as making decisions on which features in the text to use.

As described, both the Reuters and TREC FR88 documents use SGML tagging. This tagging is used to pre-process the documents into a single representation for the experiments. This representation is based upon using individual files to represent each document. Each file is named after the identity of the document it represents. They contain only the title and main body of text from each document without any of the SGML tags.

In making a comparison with the experiments carried out by Xu and Croft [18], the documents used for the histogram-based language models were treated separately from those used by the compression-based method. The compression-based method used the text found in each of the files, as shown in figure 3.3. The histogram-based language models, however, required the text to be further processed. This involves removing the less useful words from the text as well as ensuring all the morphological variants of a word (i.e. spell, spelling, spelled, etc…) have a single representation (these steps are known as stop word removal and stemming). Stop word removal is based upon identifying words that are in a pre-compiled list of stop words and removing them from the text. Stemming uses an implementation of the Porter Stemming algorithm [11] written by B. Frakes and C. Cox. Figure 3.4 shows an example of a document once it has been prepared for the histogram-based approach.

WOOLWORTH U.K. SAYS GROWTH PROSPECTS EXCITING
LONDON, March 24
Woolworth Holdings Plc WLUK.L which
earlier announced a 1986 pre-tax profits rise of 42 pct over
1985, said its prospects for growth were very exciting.
The profit figure of 115.3 mln stg exceeded a forecast by
some 10 pct made during the hostile bid by Dixons Group Plc
DXNS.L last year and the company said the results were a
major step towards the aim of making Woolworth the most
profitable retailing group in the U.K.
It…

Figure 3.3 The document from figure 3.1 after pre-processing for compression-based language models.

WOOLWORTH SAYS GROWTH PROSPECTS EXCITING LONDON March 24 Woolworth Hold Plc WLUK earlier announc 1986 pre tax profit rise 42 pct 1985 said prospect growth veri excit profit figur 115 3 mln stg exceed forecast 10 pct dure hostil bid Dixon Group Plc DXNS year compani said result major step aim Woolworth profit retail group
…

Figure 3.4 The document from figure 3.1 after pre-processing for histogram-based language models.

3.3.2 Seeding

The purpose of seeding is to select which documents are to be used to initialise the clusters. This is necessary as without any data in each cluster, it would be impossible to discriminate between them. Two methods were used to seed the clusters in the clustering experiments, one supervised and the other unsupervised.

The first method selects a random selection of documents from the collection, one to represent each initial cluster. Document selection is achieved by picking documents at uniform points throughout the collection. This increases the chances that the seeds will be better representatives of the various topics and time periods that the collection covers.

The second method, known as supervised learning, selects seed documents from the Reuters collection based on the relevance judgements provided. The Reuters collection is split into two equal halves, one for training and the other for testing. The training documents are used to seed the initial clusters. Each cluster represents a particular topic identified in Reuters, and then each document is added to the topics it has been judged relevant to. A document may be used to seed several clusters since it may have several relevance judgements assigned to it by Reuters. Some clusters may also not even have a seed document, as there may be no documents relevant to it that

appear in the training set. After training, the rest of the collection is clustered. Unlike most supervised learning approaches however, as each document is clustered the chosen cluster is updated by the document. This is to simulate the "buckshot" approach, where the clusters are primed using held-out training data, and the clustering process continues on the rest of the documents using these seeded clusters as the starting point. Half of the documents were selected from the entire collection to base these seeds on. All the documents relevant to a particular topic are combined and these are used to seed each of the clusters.

3.3.3 Clustering

An object-oriented framework was created to perform the clustering. This framework allows for an efficient, modular means for implementing the different representation techniques. It is based upon the *K*-Means algorithm:

1) First Pass
    1) Choose the first *k* documents as the initial clusters.
    2) Add each new document to its closest cluster.
2) Second Pass
    1) Take the results of the first pass as the initial clusters.
    2) For each document find the closest cluster and reassign it if its not already there.

This is the same clustering technique as used by Xu and Croft [18]. They had found that the second pass only gave a 3% gain in retrieval performance. As such, the clustering in this thesis is only based upon a single pass. This also benefits the restrictions on time and processing.

The two different representation techniques, histogram-based language models and compression-based language models, use different methods of representing documents and clusters and measuring the differences between them. The basic framework was implemented in Java 1.2; however, both techniques use methods that are called via the Java Native Interface from Teahan's Text Mining Toolkit [15]. This toolkit contains several methods for manipulating models, text and tables.

The histogram-based language modelling approach uses a trie data structure implemented in the Text Mining Toolkit to store words and their corresponding

frequencies. These are further wrapped in a Java object used to represent a document or a cluster in the framework. The difference between a document and a cluster represented by these objects is measured using the Kullback-Leibler divergence, as used by Xu and Croft [18].

The compression-based language models use the Text Mining Toolkit to represent clusters as models. The difference between a cluster and document is measured by the compression ratios, returned when encoding a document's text with a particular model. The smaller the compression ratio obtained using a particular model, the closer the document is considered to the cluster that the model represents.

4 Model Experiments

To help increase the understanding of compression based language models, as well as increase familiarisation with the Text Mining Toolkit [15], various experiments were carried out. These experiments gauge the ability of compression-based language models when used for information retrieval. They also assist in identifying the potential problems and opportunities of using the Text Mining Toolkit when implementing the clustering framework.

The first of these experiments looks at the compression of individual documents using the Text Mining Toolkit. The text from each document is compressed using an order 4 PPM character based model[2]. The compression ratios were recorded for each document, and then plotted in a graph. The compression ratios vary depending on the individual documents. Documents that have frequently occurring patterns in their text will compress better than documents that have very few. The compression will therefore also vary with the size of the documents; smaller documents with fewer features will have less chance of those features being repeated.

The second experiment involved using a dynamic model to compress each document. Again this used an order 4 PPM character based model; however, this time as each document was compressed, the model was dynamically updated to include the text from the documents. Compression ratios were recorded, as well as the size of the dynamic model as each document is added. This graph is expected to show that as more documents are added to the dynamic model, the compression of the documents improves. This is caused by patterns in the previously seen documents being repeated in future documents. Only the first few documents should have compression ratios comparable to those achieved from individual compression, with the remainder achieving much better compression. The graph depicting the growth of the dynamic model should also be curved, rapidly growing at first as new features are found in documents, but slowing down as less and less unique features are seen in each new document.

The next two sections look at the results that were achieved when these experiments were applied to the Reuters collection and the TREC FR88 collection.

[2] The Escape method used here is the same as that used in the toolkit (Method D).

4.1 Compression Results with the Reuters Collection

The results achieved by compressing documents from the Reuter's collection individually can be seen in figure 4.1(a). From this graph we can see that compression ratios do vary as expected. Table 4.1 shows just how varied, from the highest and lowest values. It also gives the mean average compression, and gives the 1st and 3rd quartile as an indication of the band in which the bulk of the documents were compressed. Compression of these documents using a dynamic model is shown in figure 4.1(b), and the growth of the model in 4.1(c). Significantly, The graph of the compression ratios does not follow the expected curve, with the compression ratios quickly settling into a diagonal line. Table 4.1 does however show that compression ratios are significantly better than those achieved by compressing documents individually. Also the bulk of the ratios are shown to fall between a much tighter inter-quartile range. The graph of model growth is also not nearly as curved as was expected, showing a sharp incline in growth when the first few documents are added, and then a linear growth as the rest of the collection is added.

	Lowest (bpc)	Highest (bpc)	Mean (bpc)	1st Quart (bpc)	3rd Quart (bpc)
Encoding Individually	2.51	7.19	4.77	4.16	5.33
Encoding with a Dynamic Model	0.84	5.37	1.84	1.65	1.98

Table 4.1 Compression of Reuters documents.

The vast improvement in compression using the dynamic model over the individual compression does indicate that patterns have been identified. The short period where the model grows rapidly and the compression ratios are high indicates that it is during this period that many of the common features are discovered. These features are then repeated throughout the remaining documents, hence the improvement in compression ratios over the individual compression ratios. However, apart from these base features, the linear growth of the dynamic model indicates that each of the documents contain roughly the same amount of previously unseen features.

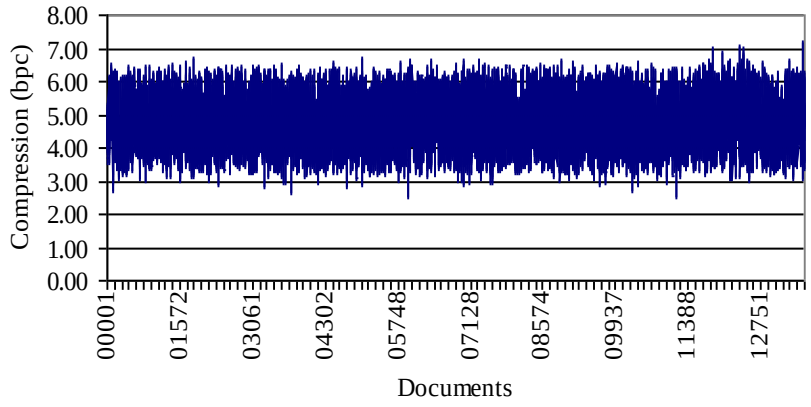

Figure 4.1(a) Encoding Reuters documents individually.

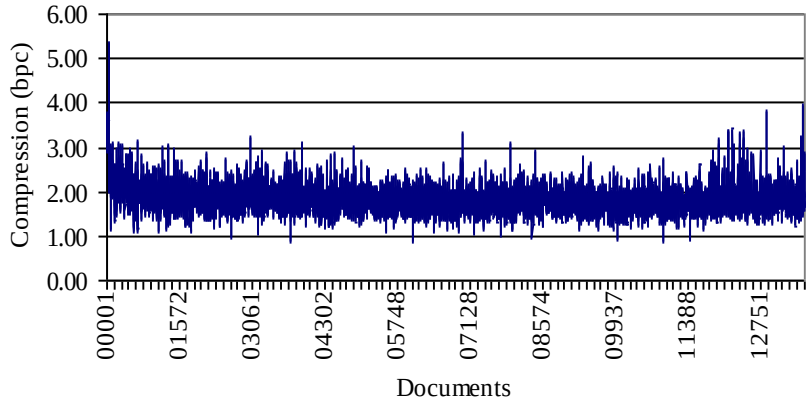

Figure 4.1(b) Encoding Reuters documents using a dynamic model.

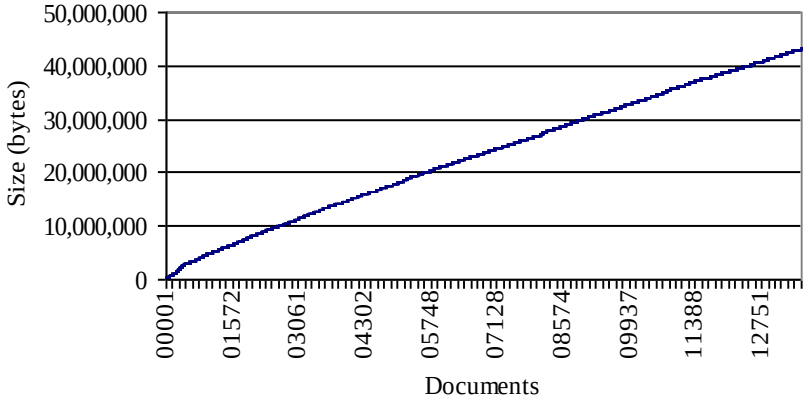

Figure 4.1(c) Growth of the dynamic model.

4.2 Compression Results with the TREC FR88 Collection

The results achieved from compressing documents from the TREC FR88 collection individually can be seen in figure 4.2(a). This graph shows that results are varied as expected with table 4.2 showing by just how much. Although the graph seems to show a more varied range of compression ratios than was achieved with the Reuters collection, the actual bulk of documents fall within a smaller inter-quartile range. These documents also compress better than the Reuters documents. Compression of the documents from the TREC FR88 collection using a dynamic model is shown in figure 4.2(b), and the growth of the model in 4.2(c). Neither of these graphs follows the curve that they were expected to. Indeed, the graph of compression ratios even seems to get slightly worse as the first few documents are added, with the lowest ratios being achieved in those first few documents. The spikes in figure 4.2(b) are from documents that contain very little or no data. Table 4.2 does show that using a dynamic model does significantly improve the compression ratios, and does put the majority of the documents within a much tighter band. The graph showing the growth of the dynamic model is almost entirely linear, only broken by the occasional small jump in growth.

	Lowest (bpc)	Highest (bpc)	Mean (bpc)	1_{st} Quart (bpc)	3_{rd} Quart (bpc)
Encoding Individually	0.85	8.79	3.22	2.61	3.69
Encoding with a Dynamic Model	0.85	8.79	1.69	1.57	1.76

Table 4.2 Compression of TREC FR88 documents.

The improvement in compression when using a dynamic model indicates that some features must be learned within the first few documents that are repeated throughout the collection. This is not so evident in the growth of the model as it was with the Reuters collection, as there is no sharp increase in new features within the first few documents. Instead there seem to be short bursts of growth throughout the dynamic model's life from the addition of documents that contain many new features.

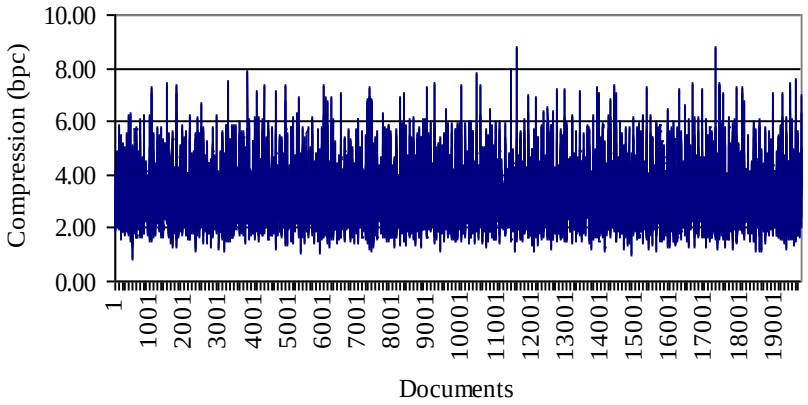

Figure 4.2(a) Encoding TREC FR88 documents individually.

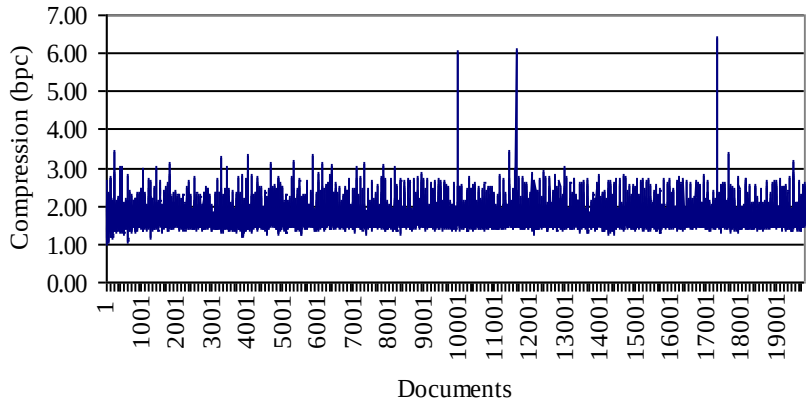

Figure 4.2(b) Encoding TREC FR88 documents using a dynamic model.

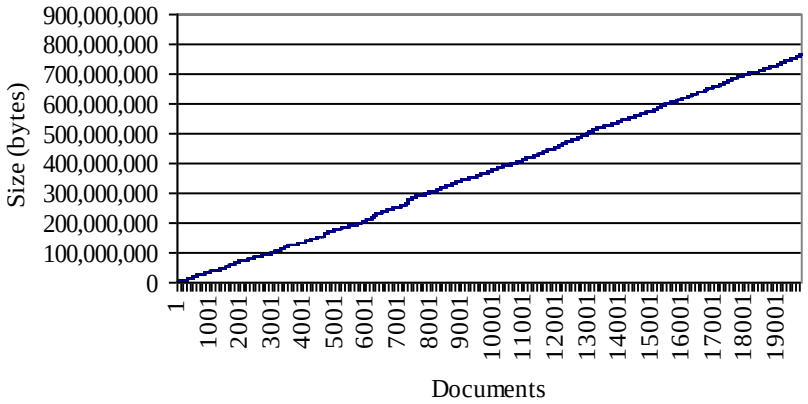

Figure 4.2(c) Growth of the dynamic model.

4.3 A Further Experiment with Reuters

A further experiment was carried out to look at text classification using compression-based language models. In this experiment, the Reuters collection was split into two equal halves, a training set and a test set. Models were created that represented each of the pre-defined categories in the Reuters collection. These models are trained upon the documents from the training set that are judged relevant to the particular topic that it represents. A document may belong to several categories, and not all categories have documents that are present in the training set. Once the models have been trained, each test document is encoded against each of the models. The compression ratios returned from encoding are then recorded.

Figure 4.3(a) shows the distribution of test documents over the categories, as judged by the relevance judgements made by Reuters. Figures 4.3(b) and (c) show the distribution of the documents as judged by classification using a character based model with a maximum order of four and a word based model with a maximum order of zero. With the word based model, when unique words are identified a character based model is used to encode them. As can be seen, the results produced by these two techniques are identical. Compared to figure 4.3(a), both techniques have performed similarly to Reuters. Some of the documents that make up the smaller categories have been lost in larger categories by the modelling techniques. Apart from this, the only other major difference is the increase in topic 1 from 17% in the Reuters judgements to 31% in both the modelling techniques. This extra 14% has obviously come from those documents that belonged to the smaller categories.

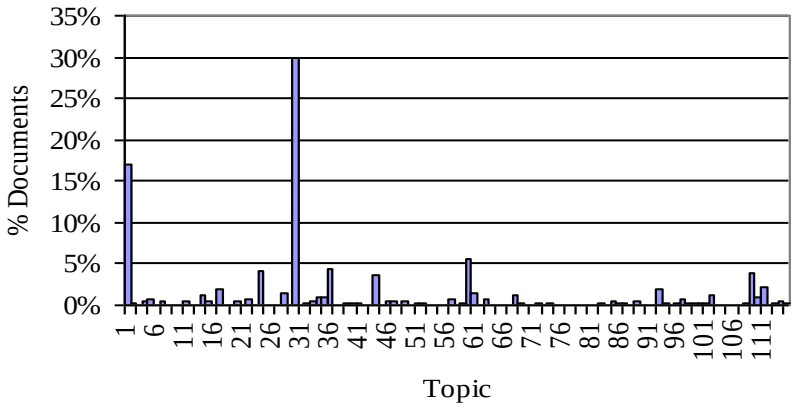

Figure 4.3 (a) Percentage of documents per topic in Reuters.

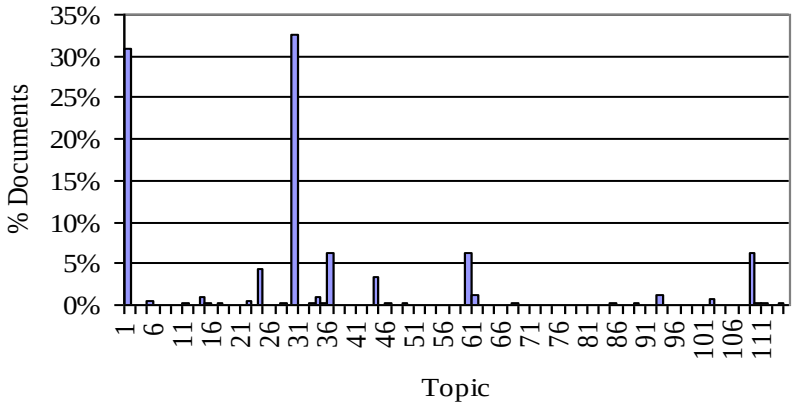

Figure 4.3(b) Percentage of documents per topic using PPMC (Character based with a maximum order of 4).

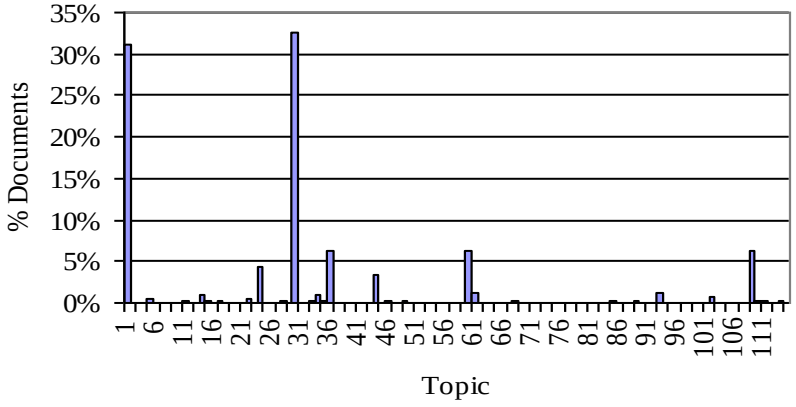

Figure 4.3(c) Percentage of documents per topic using PPMC (Word based with a maximum order of 0).

4.4 Discussion

These experiments have looked at the behaviour of compression based language models, as well as shown the ability of these models to compress and classify documents. They have thrown up some surprising results, and raise many questions that as yet have to be studied further.

The first set of experiments described in this section showed that using a pre-trained model to encode a document dramatically I mproves compression. Surprisingly however this effect is seen almost immediately when the models only have a small amount of training data. It was expected that the compression ratios would improve over time as the model grew in size. It is also interesting to note the difference in the spread of the graphs between compressing documents individually and using a dynamic model. The model growth has also shown a surprisingly constant growth, with only slight bumps where larger documents have been added.

In the second set of experiments an insight is given into how the compression-based language models will act in the clustering experiments. They have shown that these models can effectively classify documents into categories when there is an adequate amount of training data. However, the models have behaved in several unexpected ways, the reasons for which can only be investigated with further time. There are several other experiments that could have been carried out had time permitted. For example, there could have been an investigation into how well Reuters documents would compress on models trained with various data. This could have involved training the model on a standard English text such as the Brown corpus, or perhaps on text from the TREC FR88 collection. The Reuters documents could have then been compressed using these various models to see which of them gave the best compression.

5 Results

This section reviews the results that were achieved by the various combinations of representation and seeding techniques used for clustering the Reuters collection. It starts by outlining two of the techniques that were used to analyse the results achieved by the six clustering techniques. It then goes on to look at the results that were achieved and the conclusions drawn from analysis. Finally, the section concludes with a discussion and comparison of the various techniques.

The first analysis method is to use a histogram showing the distribution of the documents over the clusters. This method provides a quick means for getting an impression of how well the technique has performed at clustering. Ideally, clusters should be varied in size. No single cluster should contain the bulk of the documents, but neither should documents be evenly spread over the clusters.

The second method creates a confusion matrix of *E*-measures [12] for the ten largest categories in Reuters. The rows are made up of the top ten categories identified in Reuters, and the columns are clusters that are selected to represent the ten categories. To choose the cluster that will represent a category, the *E*-measures are calculated for all the clusters over that particular category. The cluster that achieves the lowest *E*-measure becomes the representative.

The *E*-measure is a combination of the recall and precision values. Recall is calculated as:

$$R = \frac{|A \cap B|}{|A|}$$

where A is the distribution of documents by category and B is the distribution of documents by cluster. It is a measure of the number of relevant documents in a cluster over the total number of relevant documents. Precision is calculated as:

$$P = \frac{|A \cap B|}{|B|}.$$

This equation measures the number of relevant documents in the cluster over the total number of documents in the cluster. These two formulas are combined into the *E*-

measure, which uses a third parameter α to give additional weighting to either the recall value or the precision value, depending on how important they seem. The *E*-measure is calculated as:

$$E = 1 - \frac{1}{\alpha\left(\frac{1}{P}\right) + (1-\alpha)\frac{1}{R}}.$$

For analysing these results, no additional weighting is given to the recall or precision values, so α = ½ is used. The lowest *E*-measure in the confusion matrix is ideally achieved by the row that represents the same category as the column. The lowest values in each cluster are indicated in bold. Ideally, if all the row and column categories do achieve lowest values where they cross, the matrix will have a distinct diagonal of bold values. These values will also hopefully be significantly lower than the other values in the cluster. This will indicate that the cluster contains a large portion of the documents relevant to the category that it represents.

5.1 Histogram-based Models, Random Seeds

This section evaluates the performance of the histogram-based language models with random seed selection.

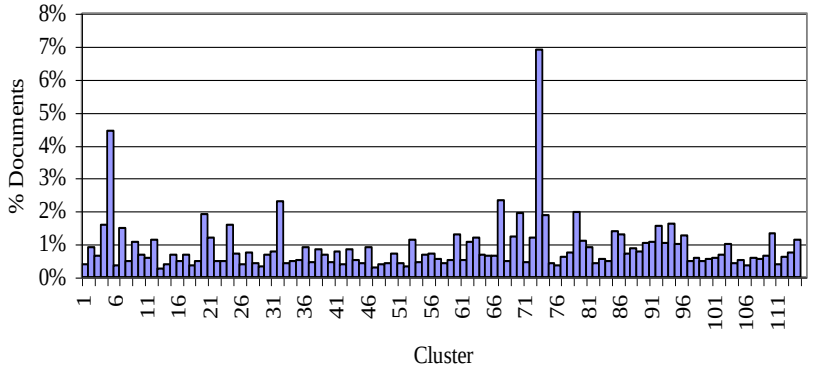

Figure 5.1 Distribution of documents over clusters.

Figure 5.1 shows a good variation in the size of the clusters, indicating that this clustering technique has worked well. It shows that the bulk of documents have not ended up in a single cluster, nor have the documents just been spread evenly over the clusters.

Eight of the ten categories have individual clusters to represent them. The remaining two - *grain* and *wheat* - are both represented by cluster 114. The confusion between these two categories is not too surprising, as it is a problem already identified by Frank, Chui and Witten [2]. They highlight the difficulties in distinguishing between documents in the *corn*, *grain* and *wheat* categories, as there are very few key features that can be used to differentiate between them.

Table 5.1 shows the confusion matrix of *E*-measures. The values for nine of the ten clusters are lowest for the category they represent. Of those nine, *crude, earn, interest, money-fx, ship* and *trade* are all significantly lower. The only cluster that does not have the highest *E*-measure for the category it represents is *wheat*, again due to its similarity with *grain*.

	acq	**corn**	**crude**	**earn**	**grain**	**int.**	**mon.**	**ship**	**trade**	**wheat**
	73	19	35	5	114	7	53	17	56	114
acq	**0.870**	1.000	1.000	1.000	1.000	1.000	0.999	1.000	0.986	1.000
corn	0.948	**0.847**	1.000	1.000	0.876	1.000	1.000	0.992	1.000	0.876
crude	0.932	1.000	**0.815**	0.998	0.993	1.000	1.000	0.911	1.000	0.993
earn	0.924	0.998	0.999	**0.743**	1.000	0.999	0.999	0.999	1.000	1.000
grain	0.937	0.861	0.995	1.000	**0.811**	1.000	0.996	0.982	0.991	**0.811**
interest	0.919	1.000	1.000	1.000	1.000	**0.585**	0.811	1.000	1.000	1.000
money-fx	0.921	1.000	1.000	1.000	1.000	0.847	**0.722**	1.000	0.997	1.000
ship	1.000	0.987	0.992	1.000	0.951	1.000	0.994	**0.657**	0.993	0.951
trade	0.949	1.000	1.000	1.000	0.997	1.000	1.000	1.000	**0.772**	0.997
wheat	0.970	0.963	1.000	1.000	0.815	1.000	0.994	1.000	0.992	0.815

Table 5.1 Confusion matrix of *E*-measures.

5.2 PPMD Character Order 4 Models, Random Seeds

This section evaluates the performance of the compression-based language modelling technique using the PPMD character based compression scheme with a maximum order of 4, with initial clusters created using random seed selection.

Figure 5.2 shows that the majority of documents have been grouped into a single large cluster, indicating that this clustering technique has performed poorly.

Nine of the ten largest categories have all been placed into cluster 87. The only other category that has a significant number of documents outside of this cluster is *earn*, represented by cluster 94. The reason that most documents are grouped into cluster 87 is because of its size. With compression-based language modelling, the more text that a model is trained on the better it will compress. These results, combined with the

results achieved in preliminary experiments involving Reuters, suggests that a model can quickly learn the language that is used throughout the Reuters documents. This one large cluster seems to have quickly become the best representative of the language, leading to better document compression. This in turn leads to more documents being added to the model, further improving its compression.

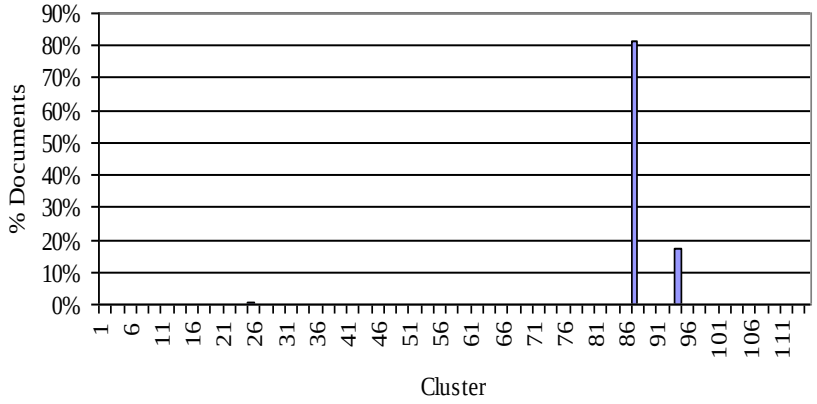

Figure 5.2 Distribution of documents over clusters.

Table 5.2 shows the confusion matrix of *E*-measures. The values for only two of the ten clusters are significantly lower than the other values for the category that they represent. These are the *acq* and *earn* categories. The reason that the *acq* category achieves the lowest value is that it is the largest of the categories represented in cluster 87. However the extremely low value that is achieved by the *earn* category in cluster 94 is of interest.

	acq 87	**corn** 87	**crude** 87	**earn** 94	**grain** 87	**int.** 87	**mon.** 87	**ship** 87	**trade** 87	**wheat** 87
acq	**0.654**	**0.654**	**0.654**	0.999	**0.654**	**0.654**	**0.654**	**0.654**	**0.654**	**0.654**
corn	0.956	0.956	0.956	1.000	0.956	0.956	0.956	0.956	0.956	0.956
crude	0.906	0.906	0.906	0.999	0.906	0.906	0.906	0.906	0.906	0.906
earn	0.776	0.776	0.776	**0.271**	0.776	0.776	0.776	0.776	0.776	0.776
grain	0.897	0.897	0.897	1.000	0.897	0.897	0.897	0.897	0.897	0.897
interest	0.917	0.917	0.917	0.998	0.917	0.917	0.917	0.917	0.917	0.917
money-fx	0.873	0.873	0.873	0.999	0.873	0.873	0.873	0.873	0.873	0.873
ship	0.952	0.952	0.952	1.000	0.952	0.952	0.952	0.952	0.952	0.952
trade	0.911	0.911	0.911	0.999	0.911	0.911	0.911	0.911	0.911	0.911
wheat	0.949	0.949	0.949	1.000	0.949	0.949	0.949	0.949	0.949	0.949

Table 5.2 Confusion matrix of *E*-measures.

A closer look reveals that the documents that make up cluster 94 are all quite small and very similar. An example of these documents is shown in figure 5.3. These

documents contain a small number of features that are unique to their type. They achieve extremely good compression when encoded using a cluster that is seeded on one of them. The larger cluster does not compete with such a cluster due to the relatively large amount of unique words in these documents.

TONKA CORP TKA RAISES DIVEDEND
MINNETONKA, MINN., Feb 26 –

Qtly div two cts vs. 1.7 cts
Pay March 26
Record March 12

Figure 5.3 Example of a document from cluster 94.

5.3 PPMD Word Order 0 Models, Random Seeds

This section evaluates the performance of the compression-based language modelling technique using the PPMD word based compression scheme with a maximum order of 0, with initial clusters created using random seed selection.

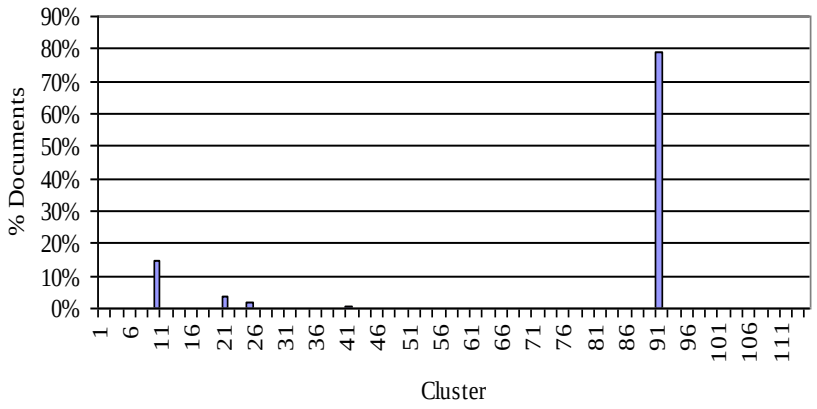

Figure 5.4 Distribution of documents over clusters.

Figure 5.4 shows the distribution of documents over the clusters. The distribution is similar to that created by the PPMD character based approach, with the majority of the collection being placed into a single cluster.

Eight of the ten largest categories have all been placed into cluster 91. The other two categories, *earn* and *interest*, have individual clusters to represent them. Despite the similarities with the PPMD character based approach, it is interesting to note that the actual clusters chosen to represent the categories are different. Even though the same

seed documents were used, there is obviously sufficient difference in the way these two methods work to cause the differences in clustering.

Table 5.3 shows the confusion matrix of *E*-measures. The values for three of the ten clusters are significantly lower than the other values for the category that they represent. The *acq* category performs the best in cluster 91 because it is the largest category represented in that cluster. The cluster representing *earn* achieves a very low *E*-measure against its category, again due to the small unique documents that make up the majority of that cluster.

	acq 91	**corn** 91	**crude** 91	**earn** 10	**grain** 91	**int.** 25	**mon.** 91	**ship** 91	**trade** 91	**wheat** 91
acq	**0.647**	**0.647**	**0.647**	1.000	**0.647**	0.994	**0.647**	**0.647**	**0.647**	**0.647**
corn	0.954	0.954	0.954	1.000	0.954	1.000	0.954	0.954	0.954	0.954
crude	0.904	0.904	0.904	1.000	0.904	1.000	0.904	0.904	0.904	0.904
earn	0.800	0.800	0.800	**0.341**	0.800	0.998	0.800	0.800	0.800	0.800
grain	0.893	0.893	0.893	1.000	0.893	1.000	0.893	0.893	0.893	0.893
interest	0.922	0.922	0.922	0.998	0.922	**0.873**	0.922	0.922	0.922	0.922
money-fx	0.873	0.873	0.873	1.000	0.873	0.958	0.873	0.873	0.873	0.873
ship	0.950	0.950	0.950	1.000	0.950	1.000	0.950	0.950	0.950	0.950
trade	0.910	0.910	0.910	0.999	0.910	0.976	0.910	0.910	0.910	0.910
wheat	0.946	0.946	0.946	1.000	0.946	1.000	0.946	0.946	0.946	0.946

Table 5.3 Confusion matrix of *E*-measures.

5.4 Histogram-based Models, Supervised Learning

This section evaluates the performance of the histogram-based language models with supervised learning seed selection.

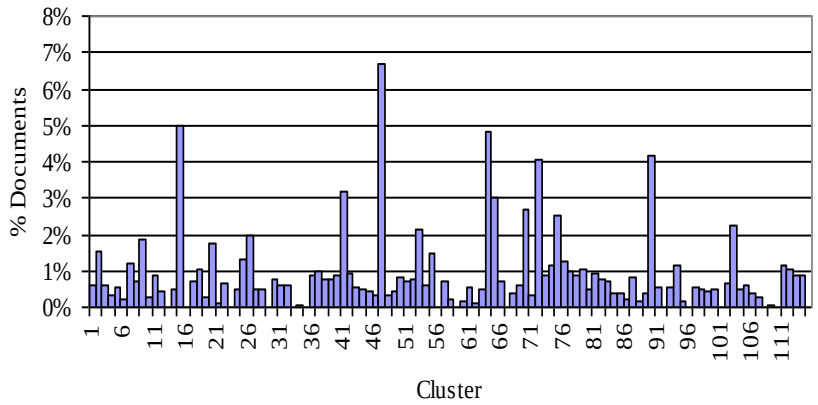

Figure 5.5 Distribution of documents over clusters.

Figure 5.5 shows a good variation in the size of clusters, indicating that this clustering technique has worked well. The variation in sizes is greater than that achieved by using even seeds with this representation method, suggesting an improvement in performance by using these seeds.

Eight of the ten categories have all got individual clusters to represent them. The remaining two – *grain* and *wheat* – are both represented by cluster 111. Again this confusion is due to the similarities previously identified between documents of the *corn*, *grain* and *wheat* categories.

Table 5.4 shows the confusion matrix of *E*-measures. The values for nine of the ten clusters are lowest for the category they represent. Of these nine, *acq*, *crude*, *earn*, *interest*, *money-fx*, *ship* and *trade* are all significantly lower. The only cluster that does not have the highest *E*-measure for the category it represents is *grain*, again due to its similarity with *wheat*.

	acq	**corn**	**crude**	**earn**	**grain**	**int.**	**mon.**	**ship**	**trade**	**wheat**
	75	81	32	64	111	43	113	82	36	111
acq	**0.855**	0.996	0.993	0.959	0.991	0.996	0.997	0.992	0.994	0.991
corn	1.000	**0.909**	0.992	0.981	0.926	0.992	0.993	0.972	0.993	0.926
crude	0.994	0.981	**0.895**	0.976	0.993	0.996	0.988	0.947	0.992	0.993
earn	0.970	0.993	0.993	**0.798**	0.988	0.996	0.998	0.997	0.995	0.988
grain	0.994	0.938	0.996	0.980	0.884	0.992	0.993	0.974	0.981	0.884
interest	0.983	0.996	0.995	0.988	0.992	**0.862**	0.991	0.987	0.968	0.992
money-fx	0.987	0.988	0.990	0.992	0.986	0.983	**0.847**	0.991	0.981	0.986
ship	0.977	0.988	0.992	0.988	0.995	1.000	0.981	**0.845**	1.000	0.995
trade	0.997	0.988	0.986	0.981	0.985	1.000	0.984	0.983	**0.817**	0.985
wheat	0.991	0.955	1.000	0.991	**0.869**	0.993	0.994	0.981	0.987	**0.869**

Table 5.4 Confusion matrix of *E*-measures.

5.5 PPMD Character Order 4 Models, Supervised Learning

This section evaluates the performance of the compression-based language modelling technique using the PPMD character based compression scheme with a maximum order of 4, with initial clusters created using supervised learning.

Figure 5.6 shows that there is some variation in the distribution of documents over the clusters. This indicates an improvement using this over when it used random seeding; however, it is still far from a good distribution.

Seven of the ten categories have all got individual clusters to represent them. The remaining three – *corn*, *grain* and *wheat* – are represented by cluster 36. Again this confusion is due to the similarities previously identified between documents of the *corn*, *grain* and *wheat* categories.

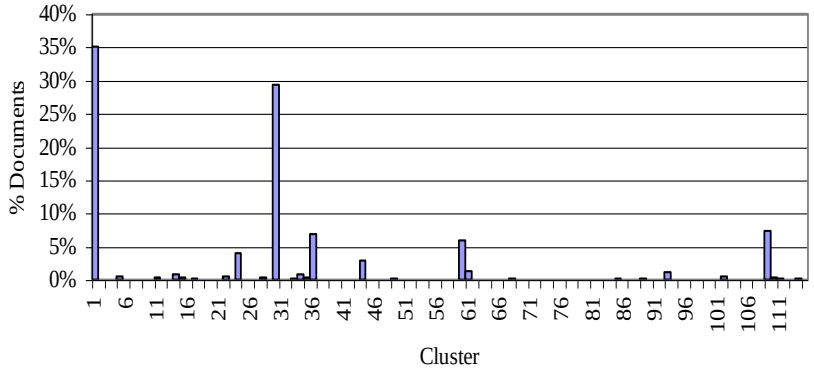

Figure 5.6 Distribution of documents over clusters.

Table 5.5 shows the confusion matrix of *E*-measures. The values for eight of the ten clusters are lowest for the category they represent. All eight are significantly lower than the other values in their respective clusters. The only clusters that do not have the lowest *E*-measure for the category they represent are *corn* and *wheat*, again due to the similarities between documents from these categories.

	acq	**corn**	**crude**	**earn**	**grain**	**int.**	**mon.**	**ship**	**trade**	**Wheat**
	1	36	24	30	36	44	60	93	109	36
acq	**0.492**	0.961	0.965	0.879	0.961	0.986	0.960	0.988	0.954	0.961
corn	0.973	0.864	0.997	0.984	0.864	0.987	0.988	0.994	0.991	0.864
crude	0.944	0.975	**0.606**	0.968	0.975	0.982	0.975	0.969	0.961	0.975
earn	0.789	0.960	0.972	**0.328**	0.960	0.982	0.968	0.989	0.951	0.960
grain	0.953	**0.726**	0.979	0.967	**0.726**	0.978	0.976	0.968	0.961	**0.726**
interest	0.958	0.982	0.974	0.975	0.982	**0.669**	0.851	0.992	0.964	0.982
money-fx	0.947	0.970	0.971	0.970	0.970	0.872	**0.698**	0.991	0.898	0.970
ship	0.969	0.968	0.932	0.985	0.968	0.983	0.972	**0.659**	0.984	0.968
trade	0.965	0.972	0.980	0.973	0.972	0.979	0.942	0.984	**0.679**	0.972
wheat	0.975	0.833	0.984	0.989	0.833	0.972	0.979	0.988	0.976	0.833

Table 5.5 Confusion matrix of *E*-measures.

5.6 PPMD Word-based Order 0 Models, Supervised Learning

This section reviews the performance of the compression-based language modelling technique using the PPMD word based compression scheme with a maximum order of 0, with initial clusters created using supervised learning.

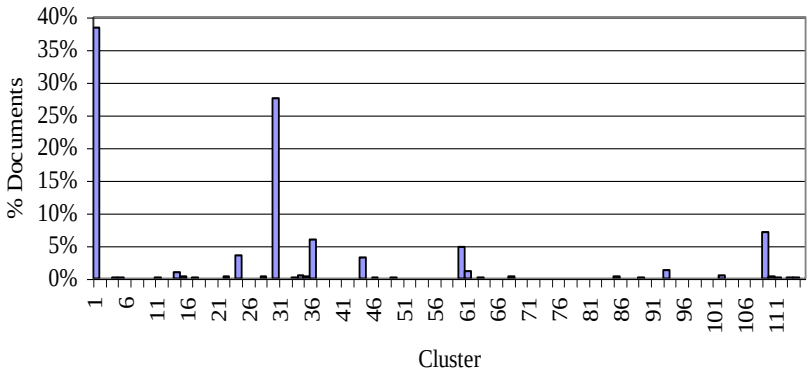

Figure 5.7 Distribution of documents over clusters.

Figure 5.7 shows that there is some variation in the distribution of documents over the clusters. This indicates an improvement using this over when it used random seeding; however, it is still far from a good distribution.

	acq	**corn**	**crude**	**earn**	**grain**	**int.**	**mon.**	**ship**	**trade**	**wheat**
	1	71	47	59	71	87	87	47	67	71
acq	**0.495**	0.964	0.970	0.878	0.964	0.982	0.982	0.970	0.992	0.964
corn	0.973	0.863	0.996	0.983	0.863	0.988	0.988	0.996	1.000	0.863
crude	0.940	0.973	**0.597**	0.967	0.973	0.991	0.991	**0.597**	0.995	0.973
earn	0.745	0.965	0.969	**0.334**	0.965	0.975	0.975	0.969	0.996	0.965
grain	0.951	**0.712**	0.981	0.964	**0.712**	0.975	0.975	0.981	0.995	**0.712**
interest	0.952	0.983	0.975	0.976	0.983	**0.620**	**0.620**	0.975	0.976	0.983
money-fx	0.944	0.969	0.961	0.965	0.969	0.773	0.773	0.961	0.983	0.969
ship	0.968	0.968	0.931	0.984	0.968	0.983	0.983	0.931	1.000	0.968
trade	0.958	0.964	0.967	0.973	0.964	0.946	0.946	0.967	**0.932**	0.964
wheat	0.974	0.822	0.982	0.989	0.822	0.973	0.973	0.982	0.992	0.822

Table 5.6 Confusion matrix of *E*-measures.

Only four of the ten categories have got individual clusters to represent them. Three of the other categories – *corn*, *grain* and *wheat* – are represented by cluster 71. Another two – *interest* and *money-fx* – are represented by cluster 87, and the remaining – *crude* and *ship*, are in cluster 47. The confusion between the *corn*, *grain* and *wheat* categories is due to the similarities previously identified between

documents of these categories. For the other categories, there are likely similarities between the documents that have made this method a poor discriminator.

Table 5.6 shows the confusion matrix of *E*-measures. The values for six of the ten clusters are lowest for the category they represent. Only the *trade* category is not significantly lower than other values in its cluster. In the clusters representing *corn* and *wheat*, the *grain* category achieves the lowest *E*-measure. Again this is due to the similarities between documents from these categories. The *money-fx* achieves its lowest *E*-measure with the *interest* category, as does the *ship* category with *crude*.

5.7 Discussion

This section has reviewed the results achieved by the six combinations of representation and seeding techniques. The performance of each of these combinations has been analysed by the distribution of documents and the *E*-measures achieved by the ten largest categories.

The histogram-based language modelling approach performs the best, indicated by the healthy distribution of documents over the clusters it creates. Both seeding techniques perform well with this technique; each giving a good spread of documents over variously sized clusters. The compression-based language models however perform extremely poorly with random seeds, with the major bulk of documents being grouped into a single cluster. Using the seeds from supervised learning does improve the performance of these representation techniques, however still does not create a distribution of documents that look as healthy as that achieved by the histogram-based approach.

Table 5.7(a) compares the performance of the three representation techniques when random seed selection was used. For each method, the table shows the E-measures that were achieved by the cluster that represents each category. Indicated in bold is the lowest of the three values for each category. This table shows that the histogram-based modelling approach is significantly better than the other two compression-based techniques, achieving the lowest *E*-measures for eight of the ten categories. By far the lowest *E*-measure in this table is achieved by the *earn* category. This is due to the small size and the unique features of many of the documents that are relevant to it.

Table 5.7(b) compares the performance of the three representation techniques when supervised learning seed selection was used. It shows that compression-based techniques seem to perform better than the histogram-based technique. However, this is only the case for the ten clusters that represent the top ten categories. A further investigation of the other 105 clusters shows that the histogram-based approach significantly outperforms the other techniques. It is due to the small number of large clusters created by the compression-based techniques that cause them to perform so well with the ten largest categories. A closer investigation of the ten largest clusters produced in these techniques shows that eight of them contain a relatively high percentage of documents from one of the top ten categories. This does indicate that these techniques have managed to distinguish between these categories with success. However, documents from the smaller categories are lost amongst these large categories.

	Random Seeds		
	Histogram	**TMTD C4**	**TMTD W0**
acq	0.870	0.654	**0.647**
corn	**0.847**	0.956	0.954
crude	**0.815**	0.906	0.904
earn	0.743	**0.271**	0.341
grain	**0.811**	0.897	0.893
interest	**0.585**	0.917	0.873
money-fx	**0.722**	0.873	0.873
ship	**0.657**	0.952	0.950
trade	**0.772**	0.911	0.910
wheat	**0.815**	0.949	0.946

Table 5.7(a) *E*-measures of the various representation techniques for random seeds.

	Supervised Learning		
	Histogram	**TMTD C4**	**TMTD W0**
acq	0.855	**0.492**	0.495
corn	0.909	0.864	**0.863**
crude	0.895	0.606	**0.597**
earn	0.798	**0.328**	0.334
grain	0.884	0.726	**0.712**
interest	0.862	0.669	**0.620**
money-fx	0.847	**0.698**	0.773
ship	0.845	**0.659**	0.931
trade	0.817	**0.679**	0.932
wheat	0.869	0.833	**0.822**

Table 5.7(b) *E*-measures of the various representation techniques for supervised learning.

Comparing the results between the two seeding methods shows a uniform improvement in the compression-based language modelling approaches by using supervised learning seeds. For the histogram-based approach, there appears to be degradation in the performance by using seeds selected by supervised learning. This, however, is not the case. Further investigation of this technique over all 115 clusters shows that it actually performs better overall using seeds selected by supervised learning.

6 Conclusions and Future Work

This report has looked at how well compression-based language models can perform at clustering text documents. This is of importance to distributed information retrieval systems, where processing is ideally focused on groupings of documents that are relevant to a user's information need. This report has also compared the performance of these models with that achieved by histogram-based language models, as used by Xu and Croft [18]. It has shown that although compression-based language models do not perform as well as histogram-based models, they do have the potential to perform better if feature selection can be improved. The remainder of this section summarises the merits of compression-based language models. It gives conclusions that can be drawn from the experiments that were carried out during this research, and finishes with a look at future work that could improve the performance of compression-based language models for clustering.

Various experiments were undertaken to measure the clustering performance under a variety of conditions. Six combinations of three different representation techniques and two seeding techniques were used. The resultant clusters were analysed by looking at the distribution of documents over them and creating confusion matrixes for the ten largest Reuters categories. The results show that when considering all 115 Reuters categories, the histogram-based language models perform the best. They give a healthy distribution of documents over the clusters and manage to group many of the documents relevant to a particular category within one cluster. The compression-based language modelling techniques do not perform so well. For random seeding, they perform extremely poorly, placing most of the documents into a single cluster. The majority of those that remain are placed in a second cluster. A closer look at these revealed they were almost entirely documents from the *earn* category. The documents all followed a similar pattern, all were very small, and contained a relatively large number of features unique to their type. By using the supervised learning seeds, the performance of these models improved. It created several clusters, each containing a significant amount of documents from one of the eight largest categories in Reuters. Indeed, when only the top ten categories are considered, the compression-based models perform extremely well compared to the histogram-based approach. However

documents from the smaller categories also end up in these larger clusters, resulting in high *E*-measures for the smaller categories.

Additional experiments were carried out during this research to look at how compression-based language models behave. The first set of experiments involved compressing documents from the Reuters and TREC collections first individually and then using a dynamic model. As expected, there is an improvement in compression when using a dynamic model, however the improvement was almost immediate and subsequently nearly constant. The growth of the dynamic model was also surprising, it continued to grow at an almost constant rate throughout the lifetime of these experiments. This combined with the instantaneous yet constant improvement in compression suggests that each of the documents in this collection contains almost equal amounts of common and unique features. A second set of experiments looked at classifying documents using compression-based language models. Documents from the Reuters collection were used to train static models that represented particular categories. The remaining documents were then encoded by each of these models, and the model that achieved best compression was noted. By comparing the number of documents judged relevant to each of the categories in Reuters with the number of documents that achieved best compression by each of the models indicates how well the remaining documents have been classified. The numbers of documents placed into each category by the compression-based techniques are similar to that achieved by the Reuters relevance judgements. These are encouraging results, as they indicate that these methods have placed the correct documents into the correct categories.

Future work is required to look at various improvements to compression-based language models to improve their ability at identifying the small but important differences between two pieces of text. There are also variations of the design issues that could still be explored, as well as further investigation of compression-based language models and their behaviour.

Further investigation of the behaviour of compression-based language models is needed to determine why they did not perform as expected. Experiments using a larger corpus of text, combined with a more detailed analysis of the results achieved could help highlight important issues. Additional experiments are also required to look at how well compression works when using models trained on different forms of

text, such as encoding TREC FR88 documents with models trained on Reuters documents.

Another modification might include basing the method for calculating the difference between document and cluster on relative entropy, as used in the Kullback-Leibler measure, rather than the cross-entropy. This measure would take into consideration global statistics over the entire collection rather than just the documents. Other possible seeding methods could also be investigated, such as using the “Buckshot” method originally outlined in the project plan. Another possible change would be to allow a document to be considered relevant to several clusters, as done by Frank, Chui and Witten [2]. This would require some method for setting a threshold of when a document should be added to a cluster. In their work, Frank Chui and Witten use positive and negative models, built from relevant and non-relevant documents. Each document is encoded using the two models, and depending on the difference will either be used to train the negative model or train the positive model.

Compression-based language models ability to detect subtle differences in documents could be enhanced by pre-processing the text in certain ways. This could involve steps similar to those taken for the histogram-based language models, where text is stemmed and stop words are removed before clustering. This would increase the pre-processing requirements for using these models; however, it would make it easier to identify the key features and trends in documents. Another possibility is to offset documents against a typical example of the language used in them. This could be achieved by encoding a document against a standard model of the language, such as the Brown corpus if the documents are written in English, and encoding it against the clusters. Another possible way to achieve this effect would be to prime all the clusters with the standard model of the language when they are being initialised.

References

[1] BERRY, M.W. Introduction to Vector Space Models (http://www.cs.utk.edu/~berry/lsi++/node4.html). 1996.

[2] FRANK, E., CHUI, C. AND WITTEN, I.H. Text categorization using compression models. University of Waikato, N.Z., 2000.

[3] GANESON, R. AND SHERMAN, A.T. Statistical techniques for language recognition: an introduction and guide for cryptanalysts. In *Cryptologia*, 17(4), pages 321-366, 1993.

[4] HIEMSTRA, D. AND DE VRIES, A. P. Relating the new language models of information retrieval to the traditional retrieval models. CTIT, Technical Report TR-CTIT-00-09, 2000.

[5] JAIN, A. AND DUBES, R. *Algorithms for clustering data.* Prentice Hall, 1988.

[6] KUKICH, K. Techniques for automatically correcting words in text. In *ACM Computing Surveys*, 24(4), pages 377-439, 1992.

[7] LEWIS, D. D. Reuters-21578 text categorization test collection (http://www.research.att.com/~lewis/reuters21678/). AT & T Labs – Research, 1997.

[8] MCCALLUM, A. AND NIGAM, K. A comparison of event models for Naïve Bayes text classification. In *AAAI-98: Workshop on learning for text categorization*, 1998.

[9] MOFFAT, A. Implementing the PPM data compression scheme. In *IEEE Transactions on Communications*, 38(11), pages 1917-1921, 1990.

[10] PONTE, J.M. AND CROFT, W.B. A language modeling approach to information retrieval. In *Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275-281, 1998.

[11] PORTER, M. F. *An algorithm for suffix stripping*. Pages 130-137, 1980.

[12] SHANNON, C. E. A mathematical theory of communication. In *Bell System Technical Journal*, 27, pages 379-423 and 623-656, 1948.

[13] SONG, F. AND CROFT, W.B. A general language model for information retrieval. In *Proceedings of the Twenty-second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkley, California, USA, 1999), pages 279-280, 1999.

[14] TEAHAN, W. J. *Modelling English Text*. D.Phil. thesis, University of Waikato, N.Z., 1998.

[15] TEAHAN, W. J. An improved interface for probabilistic models of text. The Robert Gordon University, Scotland, 2000.

[16] TEAHAN, W. J. Text classification and segmentation using minimum cross-entropy. The Robert Gordon University, Scotland, 2000.

[12] VAN RIJSBERGEN, C. J. *Information Retrieval*. 2nd ed. Butterworth & Co. 1979.

[17] VOORHEES, E. M. AND HARMAN, D. Overview of the Sixth Text REtrieval Conference (TREC-6). In *Information Processing and Management 36,* pages 3-35, 2000.

[18] XU, J. AND CROFT, W. B. Cluster based language models for distributed retrieval. In *Proceedings of the Twenty-second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkley, California, USA, 1999), pages 254-261, 1999.

Appendix A: Project Plan

A.1 Introduction

This project is going to look at techniques for clustering documents into relevant groups, and how the placement of these groups over a distributed system can have an impact on the effectiveness of distributed retrieval.

The project has been broken into four stages. The first stage involves an exploration of the clustering and language modelling software that will be used in the subsequent stages of the project. The second stage looks at various methods for selecting clusters and for representing those clusters using language models. The most effective methods determined from this stage will be selected for the third stage. It will look at what impact the organisation of the clusters will have on distributed retrieval. The fourth stage is allocated for completion of the thesis, which will also be under continual development during the project lifetime. A diary is going to be maintained throughout the project, documenting the purpose of experiments carried out and the results that are achieved. These stages will now be described in more detail.

Stage 1: Familiarisation with Language Modelling Software and Clustering.

Dates: 22/5/00 – 26/5/00

Description: Various tasks will be allocated to increase understanding of language models and clustering. Tasks will include exploration of software and development of simple applications using language models and clustering techniques.

Contingency Plan: Weekends are not included in the plan, but should be made available when necessary.

Stage 2: Comparing Clustering Techniques.

Dates: 29/5/00 – 26/6/00

Description: The *K*-Means algorithm [1] will be used with various methods for selecting initial clusters and clustering. A part of the TREC collection will be used in these experiments. There are three methods for selecting the initial clusters:

1. *Random*. This method uses a random selection of documents, each of which represents an initial cluster.
2. *Buckshot*. This method starts with a randomly selected subset of the collection. It then uses hierarchical clustering to create an appropriate number of clusters that may contain several documents in each.
3. *Supervised Learning*. This method uses training on a subset of the collection to cluster the remaining documents. Training will be achieved by sorting documents into categories according to the type of queries that they satisfy.

Four techniques will be used for clustering the documents. These techniques will model the clusters and documents and measure the similarity between them to find the cluster that a document will best fit.

1. *Topic models*. This is the technique used by Xu and Croft [1] for their cluster-based language models. In this method the Kullback-Leibler divergence function is used to measure the similarity between documents and clusters. The probability of a word appearing in a cluster is then calculated to create topic models. This can be described as a "bag of words" model, as it contains all the words that appear in a document.
2. *Word based, order 0*. This is a compression based technique, using the probabilities of words to compress the text. It is similar to the topic models in that it is a "bag of words" model, however it also includes a character based model for modelling novel words.
3. *Word based, order 1*. This technique is similar to the word based order 0 technique, except uses the current word and the word preceding it to calculate probabilities.
4. *Character based*. Again like the word based order 0 technique, but this time uses the probability of individual characters in the text.

This stage can be further broken down into the following tasks:
1. Random selection of initial clusters with each of the clustering techniques.
2. Buckshot selection of initial clusters with each of the clustering techniques.
3. Supervised learning selection of initial clusters with each of the clustering techniques.
4. Analyse and report results. The 12 sets of clusters created, (from the four techniques for each of the three tasks), will be compared by observing the placement of documents relevant to particular queries.
5. The report on the experiments carried out and there results will contribute to part of the final thesis.

Contingency Plan: Weekends are not included in the plan, but should be made available when necessary. If this stage looks likely to run out of time, then task 3 could be excluded.

Stage 3: Comparing Organisation Methods for Distributed Retrieval Systems.

Dates: 29/6/00 – 14/7/00

Description: In this stage the full TREC collection will be clustered using the initial cluster selection and language modelling technique that were most effective from the results of the last stage. The clusters will be organised using the four methods that have been identified by Xu and Croft [1]:
1. *Baseline distributed retrieval*. In this method Xu and Croft create 100 clusters based on the source of documents. The number of clusters for a source is roughly proportional to the size of the source.
2. *Global clustering*. Xu and Croft create 100 clusters for this method by clustering the entire TREC collection.
3. *Local clustering*. In this method Xu and Croft cluster each of the six TREC sources individually to create 100 clusters in total.
4. *Multiple topic representation*. For this method Xu and Croft use topic models to represent clusters of documents within ten of the natural collections of TREC. The documents are not, however, physically clustered. A topic model can only point to a collection rather than a cluster within the collection.

This stage can be broken down to the following tasks:

1. Organise clusters as in the global clustering method.
2. Organise clusters as in the local clustering method.
3. Organise clusters as in the baseline distributed retrieval method.
4. Organise clusters as in the multiple topic representation method.
5. Compare the methods with each other. The four methods will be compared by observing the placement of documents relevant to particular queries.
6. Compare the methods with Xu and Croft's results. This task will only be carried out if the project is ahead of schedule. Software will have to be developed to allow the collections to be searched in each of these methods.

This order of tasks has been chosen so that the methods that Xu and Croft found to be the most successful are carried out first.

Contingency Plan: Weekends are not included in the plan, but should be made available when necessary. Task 4 could be excluded if time is short, also reducing the time taken for tasks 5 and 6. If there is still doubt that this stage will not be completed on time, then task 3 could also be missed, again reducing the time taken for tasks 5 and 6. These tasks and there methods are specifically chosen for exclusion as Xu and Croft have shown them to be poor performers compared to the other two methods.

Stage 4: Completion of Thesis

Dates: 17/7/00 – 4/8/00

Description: This stage has been allocated for the completion and revision of the thesis. The actual thesis will be constantly under development during the projects lifetime, and there is time allocated during stage 2 for the initial write up of the experiments carried out during that stage.

A.2 References

[1] XU, J. AND CROFT, W. B. Cluster based language models for distributed retrieval. In *Proceedings of the Twenty-second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkley, California, USA, 1999), pages 254-261, 1999.

Appendix B: Literature Survey

B.1 Abstract

Distributed systems provide an efficient means for accessing large, dynamic information sources such as the Internet. The modular nature of such systems allow them to scale well with the explosive growth of information, as well as providing much flexibility over how they can be organised.

This literature survey reviews the motivation behind using distributed information retrieval systems. It looks at some of the techniques used for modelling a collection in distributed systems, as well as how these can be modified for distributed collections. It goes on to look at how distributed information retrieval systems can be organised, and reviews several of the systems that have been created. It then concludes with a summary of the literature.

B.2 Introduction

This is a survey of the research and developments that are taking place in distributed information retrieval. It looks at possible solutions to the information retrieval problem, stated by Van Rijsbergen [21] as:

> "vast amounts of information to which accurate and speedy access is becoming ever more difficult."

The World Wide Web is an example of a vast information source that poses significant challenges to information retrieval [17] [25]. It is a very dynamic collection, doubling in size every 6 months with content that is constantly under change. A study [17] has shown that several popular search services providing access to the Web are unstable, giving different results to the same query when it is submitted at different times. Over a one month period results from some of the search engines had changed by as much as 54%, compared to the estimated rate of change on the Web of 40%. This discrepancy is put down to a trade off between quality and speed. Indexes used to represent a collection can be too large to fit in a computer's main memory, so have to also be kept on secondary storage. Since accessing this can significantly slow down processing, search services try to only use the partial index in main memory at the cost of the quality and coverage of results.

The service provided to users is also affected by competition between the search services. They keep much of the underlying technology hidden from the public [25], but this means that users cannot create a good mental model of how the system works or how it will react to certain input. Witten et al. [25, page 440] have summarised the problems with current search services in this quote:

> "never in the history of information have so many depended on so few for so much – and been kept so utterly in the dark about what it is they are getting (and, more to the point, not getting)."

Distributed information retrieval systems provide a means for accessing vast, distributed collections. They provide autonomous control to the local system over

their part of the distributed collection. This allows an author to say how they want their information presented and accessed over the system.

B.2.1 Report Structure

The rest of this literature survey is organised into three sections:

- Section 2 will look at what a distributed system is, and the issues involved in creating a distributed system.
- Section 3 will look at the various techniques for modelling the resources available in a distributed information retrieval system.
- Section 4 describes how distributed systems are represented. It also looks at some of the issues involved in creating a distributed information retrieval system, and describes some systems that have been implemented.

B.3 Distributed Information Retrieval Systems

This section outlines what a distributed information retrieval system is. It starts by considering what a distributed information retrieval system is. It then looks at distributed systems, and goes on to consider the components that are required in a distributed information retrieval system.

A distributed information retrieval system provides access to a collection of documents that are spread over several sites. These sites can be geographically distant, with varying levels of control over their part of the collection. The distributed system can then provide a single access point to representing and searching all of these collections.

A distributed computer system can be defined as a *loosely coupled* or *tightly coupled* system. A loosely coupled system consists of separate computers connected together by a network. Each computer is completely autonomous, as it does not share its processor or memory with any other computer. Tightly coupled systems allow a task to be allocated over several processors, such as in processor array systems.

In this literature survey, only loosely coupled distributed systems are considered. The characteristics of these systems [19] make them an excellent solution for retrieving information from large distributed collections such as the World Wide Web. Each of the autonomous parts of the system can work in parallel over their own part of the collection. Taking advantage of the modular and parallel capabilities of these systems can create flexible, scalable solutions to the information retrieval problem. They can be organised to reflect the actual structure of the information space, placing control over the access and indexing of local documents to the local system. This autonomy is desirable as it can allow authors to control how their documents are presented to other people.

There are various components that make up an information system. Centralised information retrieval systems, such as figure 1 (a), require mechanisms for representing and searching a collection.

A distributed information retrieval system, such as figure 1 (b), requires much more thought. A distributed collection has to start with components for fragmenting and allocating documents. Components are also required for creating document and collection representations. The collection representations are used for the collection selection process, where a portion of the available collections is selected to cut down on the information space that has to be searched. The components for searching documents at each collection and for merging the results from the collections also have to be considered.

These components can then be organised in a variety of ways. In the system in figure 1 (b), all of the components are located within each retrieval system. One possible variation of this would be to allocate the collection representation, collection selection and results merging components to a mediator. Users could then access selected retrieval systems via the mediator. There are however hundreds of other potential ways for distributing these components and allocating work to them.

Decisions made on the implementation and organisation of components can depend upon the particular requirements a system has to fulfil. The effectiveness of the system can be affected by the modelling technique used to represent the distributed collections, and efficiency can be affected by how the system is organised. The level of autonomy that each local site requires over its collection, and how reliable access to that site is has to be considered. The system should also be flexible to a certain extent to cope with changes in the information space as well as changes in the underlying network.

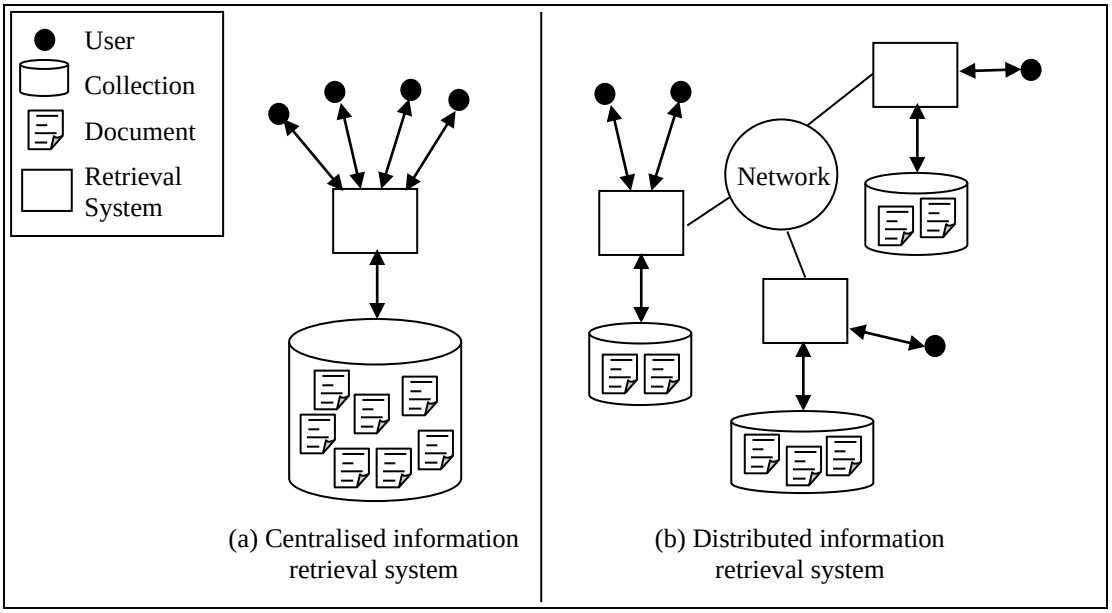

Figure 1. A centralised and a distributed information retrieval system.

B.4 Modelling Techniques

Modelling techniques are used to represent and search a distributed collection. Traditionally, in the information retrieval world, there are two distinct forms of modelling [16]. The first is abstract models, used to visualise the data without any of the implementation details. The vector space model is an example of an abstract model. The other technique is to use explanatory models, such as the 2-Poisson model, to describe data as a probabilistic mathematical model. Language models, used in speech recognition, have become a third form of representing a document in information retrieval.

This section starts with a look at two popular modelling techniques that are used in information retrieval systems. It will then focus on language models, look at modifications that have been made to the model and review a language modelling approach to distributed information retrieval.

B.4.1 Vector Space Model

In the vector space model [2] a document is represented as an *n*-dimensional vector. A typical way of implementing these models is to use a dimension for every term that appears in the document; however there are possible alternative abstractions such as phrases. Queries are also represented as *n*-dimensional vectors. The distances can then be compared between the query and documents in the information space. Documents that have similar semantic content to the query will be closer to the query in the space. No significant examples in the literature have been found of vector space modelling being used in distributed information retrieval systems.

B.4.2 *tf.idf*

The *tf.idf* function is frequently used in probabilistic modelling. A weight is assigned to each term in a document as a measure of how well that term represents the document. It is the product of the frequency of the term in the document (*tf*) and the inverse frequency of documents that contain the term (*idf*).

Callan, Lu and Croft [8] use a variation of *tf.idf* for searching a distributed collection using an inference network. Inference nets use a directed acyclic graph to represent a user's information need, query terms, documents and document terms. These nodes are connected together using weighted arcs, and it is for these weights that a variation of *tf.idf* is used.

The *tf.idf* function is modified to represent collection statistics rather than document statistics. The term frequency therefore becomes the document frequency (*df*), the number of documents that contain a particular term. The inverse document frequency becomes the inverse collection frequency (*icf*), the number of collections containing the term. To test the effectiveness of this modified function, an optimal ranking of the collections is created using the original *tf.idf* function. The ranking created by the modified *df.icf* function is then compared with this, and is found to get 75% of the rankings correct [8].

A possible reason that this is not higher is that smaller collections that contain a lot of relevant documents are being obscured by larger collections. This problem can be

overcome by extending the function model so that it takes into account the number of relevant documents found in a collection. Ranking the collections using this new model gives a 38% improvement over the older model [8]. Although this is still not achieving the optimal ranking, it does show that inference networks, using a variation of *tf.idf*, can be effective at ranking collections.

B.4.3 Language Models

Language models were originally used in the speech recognition community to model the statistical regularities in the generation of language. Ponte [15] describes a language model as:

> "a probability distribution over strings in a finite alphabet"

In information retrieval, language modelling can be used to represent the probability of a word being used to describe a document. Language models create a probability distribution $\{p_1, p_2 \ldots p_n\}$ over a vocabulary set $\{w_1, w_2 \ldots w_n\}$. For example, consider a document that after stop word removal and stemming has the following vocabulary set: {information, retrieval, distribution}. The word frequencies for these terms in the document are 2, 4 and 11 respectively. The probability distribution could then be created using a method described in [26]: {0.1180, 0.2354, 0.6465}. These scores reflect how well the words describe the document, so "information" represents 11.8% of the document, "retrieval" 23.54% and "distribution" the most descriptive at 64.65%.

There are several benefits to using language models for information retrieval over other methods [16]. Documents are modelled individually, and are not put into pre-defined classes for particular queries. There is no notion of relevance with language models. It is the probability of a document generating a query that is measured, not how relevant a document is to a query. A language model can also integrate the indexing and retrieval models into a single model. They are non-parametric, meaning they do not need any additional information to create the models. Instead they are entirely based upon the collection statistics.

Ponte and Croft [16] use language models for information retrieval by calculating a maximum likelihood estimate for each of the terms in a document. This estimate represents the probability that a term could be generated by a document. The estimates for all the terms in a query can be combined to create the probability of a document generating that query.

B.4.2.1 Language Model Modifications

The basic language model described in the previous section does however have two problems [16]. The first is that when a query term does not appear in a document, the overall probability ends up being 0. To overcome this, the average probability of the term over the entire collection is taken. The second problem is the confidence in the maximum likelihood estimate. Since documents can vary in size and content, a more robust estimate based on a larger amount of data is required. The mean probability of a term in all the documents that contain it can be used. This does however assume that all the documents are typical. To improve upon this, the frequency of a term in a document is compared to the normalised mean term frequency. In documents where a

term frequency is typical of other documents, the mean term frequency is a safe measure. The further it moves from the mean, the riskier it becomes to use.

A further improvement suggested by Ponte and Croft can be used to smooth the estimate of the average probability for terms that do not appear in many documents. Rather than estimating the average probability from the small amount of data available for a rare term, base the estimate on all the terms that are equally as rare.

Song and Croft [20] also identify the need to ensure that a term that is in the query but not in the document does not result in a zero probability. They use an estimate that allocates some probability mass to such missing terms. Further expansion of the model is used to adjust the probability of missing terms based on information about the term in the rest of the collection. Another improvement made to the model takes into consideration the sequence of query terms, useful for checking for duplicates and also for phrases. A further improvement looked at term pairing, where phrases of word pairs are used.

B.4.2.2 Cluster-based Language Models

Xu and Croft [26] define four methods for representing a distributed collection using language models. The documents are clustered to create *topics*. These topics are then represented using language modelling, and these representations are called *topic models*.

The *2-pass K means algorithm* is used for clustering related documents into groups. In the first pass, a portion of the documents is taken as the initial clusters. A distance metric is then used on each of the remaining documents to find the cluster it is closest to. The second pass takes the results of the first pass as the initial clusters, and goes through each of the documents to ensure that they are in their closest cluster. If not, then they are moved to the appropriate cluster.

The initial clusters can be selected in a variety of ways [9]. Xu and Croft simply used the first few documents. Other possibilities involve randomly selecting documents, comparing documents to find the most similar/dissimilar or using hierarchical divisions/fusions of the collection until an appropriate number of clusters are created.
Xu and Croft suggest four methods for organising the topics and topic models.

The first method is baseline distributed retrieval, as shown in figure 2 (a). Documents are not clustered into topics in this method; instead a topic model is created for each collection. This organisation represents how many existing distributed information retrieval systems currently organise their system. Collections are completely autonomous, and do not have to be changed in any way. However they are also heterogeneous, meaning that documents that will satisfy the user's information need could be spread over several sites.

The *global clustering method*, in figure 2 (b), requires that all the collections are stored at a single site. They can then be combined and clustered to create tightly bound topics. These are then modelled to create the topic models. A single topic is likely to contain all of the documents that will satisfy a user's information need.

However, the collections are centralised and there is no local autonomy over the individual parts.

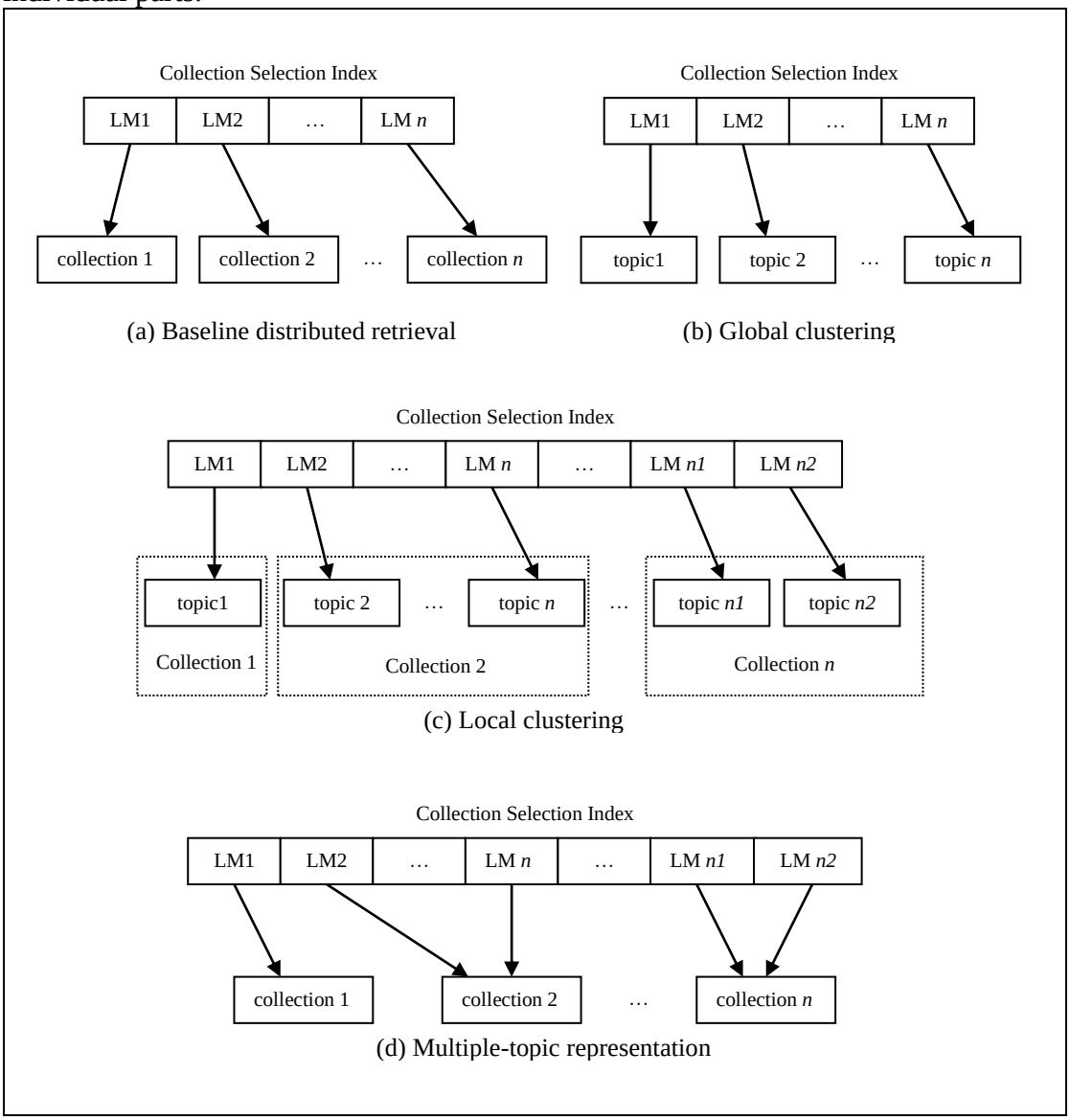

Figure 2. Four methods for organising cluster-based language models [26].

Local clustering, shown in figure 2 (c), clusters the documents in a collection to create topics, and then models these to create topic models. The collections have partial autonomy over the documents, as they will be stored at the local site. However the clustering does require control over how the documents are grouped. The topics within a collection are therefore tightly bound, but topics that could satisfy the user's information need may be spread over several collections.

The final method, *multiple-topic representation*, is shown in figure 2 (d). Like local clustering, each topic model represents a topic within a collection. However the collections have complete autonomy over documents, as they are not physically clustered. This does mean that the topic models can indicate the presence of a topic within a collection but will not have knowledge of where the particular documents that make up that topic are located. Documents that may satisfy the users need could also be spread over several collections.

Xu and Croft compare these methods with centralised retrieval [26]. Each method ranks collections for several queries. They then search the top 10 collections, retrieve 30 documents from each, and merge the results based on the document scores. The precision of the retrieved documents is then measured for various levels of recall. It was found that all the methods were improvements of the baseline distributed retrieval method, which reflected the organisation of many existing distributed information retrieval systems. The global and local clustering methods were found to be very competitive compared with the centralised retrieval method.

B.5 Architectures

This section describes the various components that are used in distributed information retrieval systems and how they can be organised. It starts with a look at how distributed systems are organised, and looks at a popular network topology that is used in many distributed systems. It will then discuss some of the design issues in creating a distributed information retrieval system. It will then describe some of the existing distributed information retrieval systems that are available.

B.5.1 Distributed Systems

The organisation of a distributed system can be described by the application architecture and logical network topology that they use. Application architectures describe the allocation of components in the system, and the network topology represents how those components are linked.

B.5.1.1 Application Architectures

The application architecture can be classified in one of three configurations dependent on where presentation, processing and data components reside [19].

In the one-tier application architecture, all the presentation, processing and data components are stored at a single location. Traditional mainframe applications are an example of one-tier applications. All the processing is done on the mainframe and clients use the application via dumb terminals that have minimal processing capabilities.

The two-tier application architecture separates the presentation, processing and data components between the server and the client. These systems can be configured as *fat client* or *fat server*. A fat client has the presentation and processing components on the client, and the data component on the server. In the fat server configuration only the presentation component is on the client, and processing and data components are on the server.

The three-tier application architecture separates the presentation, processing and data components into three locations. This is the typical architecture used by distributed information retrieval systems. In these systems, the first tier would be the front end to the system on the client's machine. The third tier would be the search engine at each of the sites, and the second tier would process the data from the search engines and present results to the client. Although called three-tier, this architecture is also used to describe applications that have several processing components between the presentation and data components.

B.5.1.2 Topologies

A network topology describes the arrangement of components and the links that they communicate through [19]. Figure 3 shows some of the topologies that are common in distributed systems.

(a) Fully connected topology, where every component can directly communicate with every other component.

(b) Partially connected topology. Indirect communication requires the message be passed along shared links. This saves on network costs.
(c) Tree connected topology, where the components are linked into a hierarchy. Variations of this topology are very popular and are used by several protocols, including Network News [10] and Inter Relay Chat [14].
(d) Star topology. This uses a central component to coordinate communication between the other components.
(e) Ring topology, where the components are linked to create a ring.
(f) Bus topology, where a single link is used to communicate between the components. This topology is ideal for group communication applications.

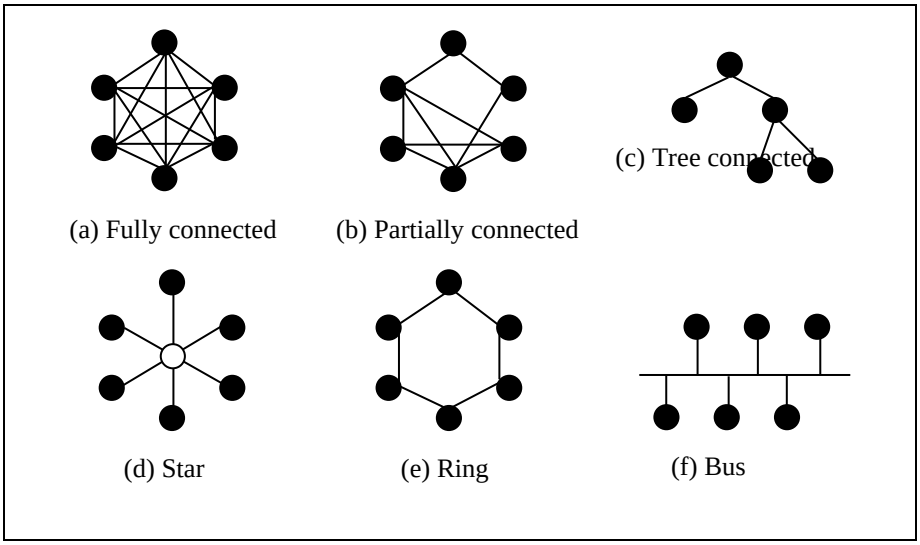

Figure 3. Network topologies [19].

B.5.2 Design Issues

There are various issues that have to be considered when creating a distributed information retrieval system. Ensuring that a system will be effective and efficient is not just a matter of selecting an appropriate modelling technique and the correct architecture. It also requires the system to be able to cope with the load put on it by various numbers of users and sizes of information collections. There are also issues of how the system will maintain a global view of the data, how to emerge the results and what replication and caching techniques can be used to improve performance.

B.5.2.1 Evaluation of Architectures

Most distributed information retrieval systems use a three-tier architecture, with a mediator allowing clients to connect to several search servers. Cahoon and McKinley [5] [6] have created a simulation of such a system. Simulators provide an effective and flexible way for looking at various performance issues in a controlled environment.

They created prototype distributed information retrieval system that uses *a central administration broker* to mediate between the clients and a group of INQUERY [7] retrieval servers. The broker places commands destined for the servers in a queue, and temporarily stores results from the servers until they have all replied. The simulator is developed based on the results of the prototype. To validate the simulator, the

prototype and a simulation of it are compared. Similar trends are observed in both systems, and as such it can be considered a good model of the real world situation.
The simulator accepts various parameters such as the number of users and INQUERY servers, query lengths and term distributions, the number of documents that match a query, the think time to represent users evaluating their results, and the number of retrieval operations that are done.

Simulations show that when searching all the INQUERY servers using small queries, adding up to 8 servers improves performance. Any more than 8 does however degrade the performance as the broker becomes over utilised. For larger queries, any more than 4 servers decreases the system performance due to the over-utilisation of the INQUERY servers. A simulation of collection selection, where random subsets of the available INQUERY servers are used for each query, yields similar results. Simulations using 2 and 4 brokers show significant improvements for the larger configurations that use many INQUERY servers.

These simulations show that for small queries using a small number of central administration brokers to manage a large, popular collection can increase the system performance. However, when long queries are used, the INQUERY servers can become a bottleneck in the system and system performance deteriorates.

B.5.2.2 Dissemination of the Collection Representation

With a collection that is distributed over multiple sites, it is necessary to ensure that all sites have an appropriate representation of the entire collection [23]. The simplest method involves a site updating its own representation when documents are added to its local collection, and then passing it on to other sites. When another site receives the new representation, it can merge it with its own. There is however a large cost in transmitting entire representations between sites. An alternative is to just transmit the updates to other sites. Each site can then recalculate its own representation based on the updates. Time stamping can be used to ensure that updates are received in order. This method does still require quite a lot of network traffic, and both of these methods require computationally intense algorithms at all the sites.

Viles [22] proposed a method for disseminating information using an additional agent, called an *administrator*. The administrator receives an update from a site, updates its representation and sends the updated representation back to the site. Whenever a site updates its representation, it will get all the other updates made by other sites. The conversation between the site and the administrator is synchronous, and therefore no time stamping is required. If a site does not have any new documents to update the representation after a certain period, it could send in an update request just to ensure that it has the latest representation of the global collection. A site can therefore update its representation in batch, and receive all the updates from other sites at the same time.

Multiple administrators could be used. This does however move the dissemination problem to the administrators. They would all have to ensure that they kept an up to date copy of the representation for distributing to the sites. However, this is still an acceptable solution as the administrators can be more dedicated to ensuring that they share the same information, and there will be a lot less of them than actual sites.

Further study by Viles and French [23] looked at how often dissemination should occur, and just how much information needs to be disseminated. They looked at two types of system for varying amounts of disseminated information. One system is based on random allocation of documents, the other on content-based allocation of documents, where documents that are relevant to the same information need are placed in the same location. Random allocation did not show significant improvements in retrieval effectiveness when information is disseminated. However content-based allocation did show much larger changes. Results also showed that optimal retrieval effectiveness could be achieved by disseminating 20% of a site's representation to all other sites.

B.5.2.3 Merging Results

When a distributed information retrieval systems has searched a number of its sites for documents relevant to a user's need, it is desirable that it creates a consolidated, consistently ordered list of results to present to the user. In various studies that have tackled this problem [1] [8], there are four distinct methods identified for combining results.

The first method is called *interleaving*. It simply takes the results from each collection and mixes them. One possible means for mixing them could be to iteratively take a single result from each source and add it to the list. This method fails when some of the selected collections actually contain few, if any, of the relevant documents. Results will then be interleaved and some of those from the irrelevant source will be higher ranked than those from other, possibly more relevant, sources.
The second method relies on the scores from each of the collection being comparable. This is known as a *raw score merge*. Comparable results cannot even be guaranteed by using the same searching mechanism at each site. If the scores given to the documents by each site is based upon term frequencies in the collection then they may vary wildly between collections.

A possible solution to the problem of incomparable scores in the raw score merge is to normalise the scores. This becomes the third method – *normalised score merge*. For example, the ranking based on term frequencies could normalise scores by taking into account global term frequency statistics.

The final method, *weighted scores*, gives each result a weight that can be based on the documents score and/or the collection's rankings. This provides an acceptable solution to the problem of ranking irrelevant document too high in interleaving. Instead, collections that contain few – if any – relevant documents will be ranked below other more acceptable collections, and the resulting weight given to the result will reflect this.

B.5.3 Distributed Information Retrieval Systems

Several distributed information retrieval systems have been created and are used for searching distributed collections. Meta-search engines are such systems that have found much use on the Internet, where they combine the efforts of several search engines into a single point. Other systems include Harvest, Discover, HyPersuit and What'sHot.

B.5.3.1 Meta-Search Engines

Meta-search engines use a three-tier application architecture, made up of the clients, the actual meta-search engine and the source search service [1]. These components are connected based on a star topology, where clients and source services are all connected via the meta-search engine.

A meta-search engine should be able identify the sources that are relevant to a user's information need, and then produce a consolidated, consistently ordered list of results combined from these sources. Figure 4 shows how the mediator interacts with the user and information sources. Details such as the various interfaces, features and formats of each of the information sources should be hidden from the user.

Informia [1] is an example of a meta-search engine. Unlike many other mediators, however, it combines formal database modelling techniques with information retrieval relevance techniques. Database modelling is used to create meta-information on the structure and content of an information source. This helps to guide the selection of appropriate sources and communication between them. Combined with relevance techniques, results do not have to be classified as a match or not. These ensure a user gets documents that could be relevant to their information need, and not just those that are an exact match to the query.

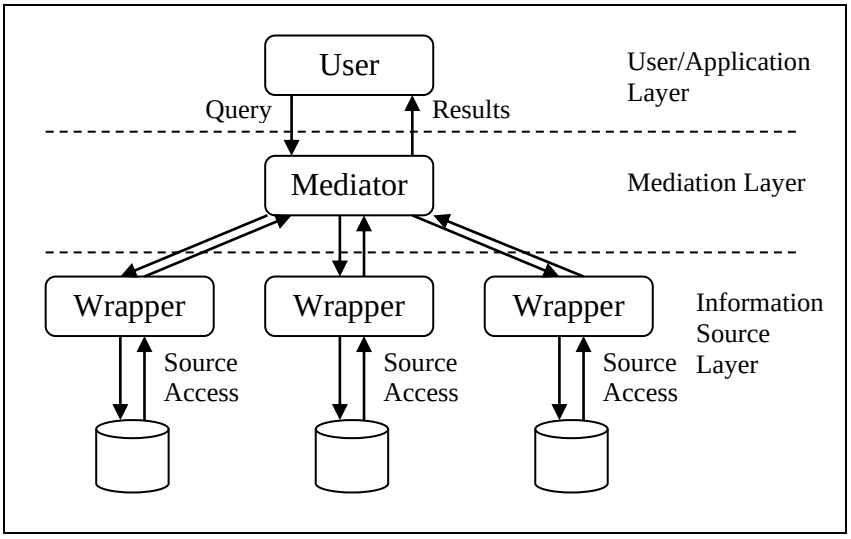

Figure 4. The meta-search engine architecture [1].

B.5.3.2 Harvest

Harvest [3] [4] [11] uses a set of customisable tools for gathering, indexing, caching and replicating information from distributed heterogeneous information sources. The Harvest system is based on a three-tier application architecture. The components are very versatile and can be connected in several configurations.

The Gatherer subsystem is used to collect indexing information. It can be used to collect from several information sources, but the most benefits can be achieved by running it at the source. A Gatherer periodically scans its information source for new and modified resources and stores content summaries in a local cache. They use a customisable information extraction system called Essence. It can be used to access a

variety of file types, including compressed and postscript files, and extract relevant information from them.

Brokers provide an indexed query interface to the collected information. Indexing information can come from one or more Gatherers or from other Brokers, enabling complex combinations to be created to maximise the efficiency of the system. A Broker manages its own index and deals with queries by invoking the Index/Search Subsystem.

A general interface between the Broker and Index/Search Subsystem has been created to allow a variety of underlying search engines to be used in the Harvest system. Currently there are two search and index subsystems that have been developed for Harvest. The first, Glimpse, is space efficient, and the second, Nebula, sacrifices space to make query resolution faster.

A replica can belong to one or more replication groups, in which a master periodically scans the group to ensure that the most efficient means of distribution is being used. Two techniques are used to maintain replicas. The first, *mirror-d*, is used to occasionally flood all of a Brokers information to a neighbouring Broker in the group. This is layered on top of the other technique, *flood-d*. Flood-d is used to pass objects on to other replicas, based upon a logical network topology designed by the group master. Where the flood-d fails to pass on updates, the mirror-d will eventually pass on all the information that has been missed.

The Object Cache is used to decrease the demand on network links and information sources. It is organised hierarchically, ideally based on the underlying network structure. When a request to get an object is received, it finds the quickest way to get the object, either from the original site, from neighbouring Object Cache, or from its parent Object Cache.

B.5.3.3 Discover

Discover [11] [18] is based on a content routing system. It is a three-tier system that uses a hierarchy topology of content routers with information sources at the leaf nodes. Each source registers with a content router, and users can query or browse that source via the router. The routers can also contain their own documents that a user can query or browse, as well as facilities for query refinement and routing. The contents of a source or router are represented in routers by a content label.

A prototype Discover system is used to build a distributed information retrieval system with access to over 500 WAIS servers [19]. This is used to validate the approach taken by Discover with a large, heterogeneous information space. The lack of control over the actual information sources does mean that content labels have to be created from the limited information that is available. In the case of WAIS servers, a catalogue file is used which provides a headline, normally a title, for each document. The information about a WAIS server is also supplemented by a short source file, which lists general information about the server and what it provides.

Results from the prototype suggested that Discover could be a useful tool for distributed information retrieval. Further enhancements to the system could improve

its time and space requirements, however the actual performance of the prototype is considered satisfactory. The exact-match searching used by Discover could probably be improved upon using other modelling techniques, and getting more statistics on information sources could also improve the retrieval effectiveness.

B.5.3.4 HyPersuit

HyPersuit [11] [24] is a three-tier application that is based on a tree topology. Documents can be clustered into a hierarchy based on some attribute, such as the document location. Each of the clusters is stored on a content router, which are connected based on the same hierarchy as the clustering. Figure 5 shows an example tree – the leaf nodes of the tree are the actual documents that are stored on leaf information servers.

To create a cluster, similar documents and clusters are fused together based on their similarity. The similarity function used has two components, one that measures the similarity of two documents based on the content of the documents, and another that is based on three notions of how the documents relate in hyperspace. The first notion looks at the direct path of hyperlinks between the two documents, giving more weight to documents that are closer together. The second looks at common ancestors that the documents share. This measurement depends upon the number of common ancestors and how close they are in terms of hyperlinks. The final notion looks at common descendents. Again the measure is affected by the number of common descendents and how close they are. These components combine to create a means for building a cluster hierarchy that closely reflects the structure of the information space.

A prototype HyPersuit system has been built that is comprised of 100 Web sites organised in a 4 level hierarchy of 42 content routers. The hierarchy is constructed to reflect the domain name system. The top level represents educational servers, the second level particular educational institutes, the next level particular departments or groups within the institutes and the fourth level represents the machines that serve Web sites in those departments. HyPersuit is useful for modelling the underlying hyperspace structure, and is considered to be scalable to any size of information space. There is not a lot of detail in the literature however on the actual effectiveness of this system as an information retrieval system.

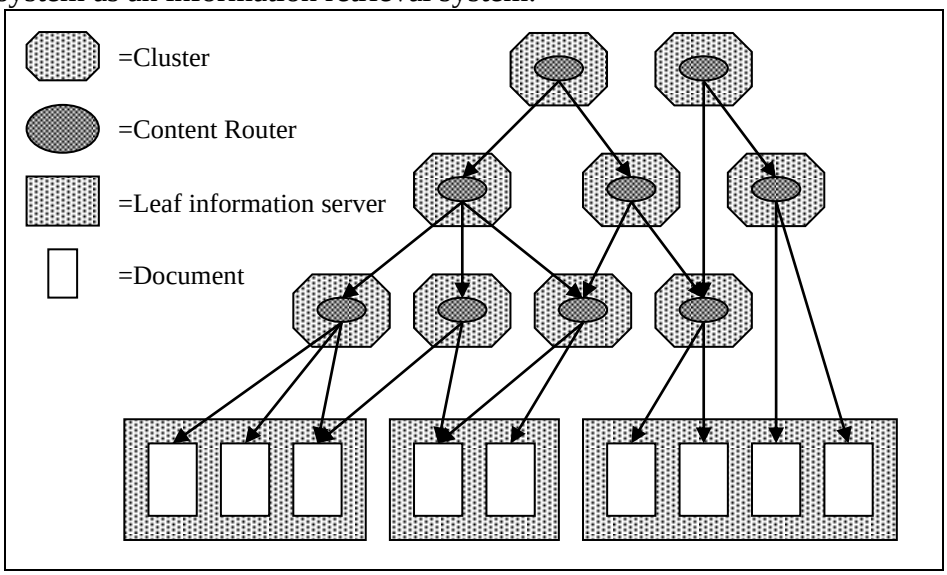

Figure 5. Content routing as cluster hierarchies [24].

B.5.3.5 What'sHot

What'sHot [11] is a three-tier application composed of index brokers. Brokers are ideally run on local networks, where they act like a cache to local users storing document representations as long as they are of interest to the local users. This system allows direct communication between the brokers, which would be linked to form a fully connected topology.

The contents of a document are summarised to create an information abstract for representing the document in the system. Abstracts are classified by a broker into one of three groups: abstracts that are popular with local users, abstracts that represent the most prominent topic in the broker, and abstracts published by local users.

The main advantage of What'sHot over other systems is that it promotes specialisation. Index brokers promote specialisation by ensuring that they keep abstracts that are relevant to their particular topic. When a user has a query on a particular topic that the local broker cannot resolve, then there will hopefully be a broker that does specialise in that topic and the query can be passed to it.

A broker becomes a specialist by reviewing its abstracts to find the most prominent topic. If there are no other brokers that specialise in that topic, then it can become the specialist for it. To ensure that it stays the specialist, the abstracts that make up the topic are given a longer life in the cache. Abstracts are also passed about between brokers; any that compliment the specialist topic can be accepted and added to the cache. When a specialist cannot resolve a query that is relevant to its specialist topic, it passes it to another broker that may be able to answer it. The resultant set of relevant abstracts that are then returned can be added to the broker's specialist topic. In these ways each broker accumulates abstracts on its particular topic. Queries can then be routed to the appropriate specialist and a set of relevant, popular results will be returned.

B.6 Summary and Conclusions

This survey has reviewed many of the developments in distributed information retrieval systems. In the introduction, it looked at the motivation behind using distributed systems for information retrieval.

Section 2 explored what a distributed system is, and has looked at some of the general issues that have to be considered about the creation of a distributed system.

Section 3 has reviewed three of the most popular modelling techniques for information retrieval, and focused on the modifications that have enabled two of them to represent distributed collections. These two techniques have shown that distributed information retrieval systems can be competitive with centralised ones. They have both shown an efficient, effective and transparent means for providing a user with access to several distributed information sources. The experiments carried out using these models have however been limited. They have been shown to be effective on large, controlled information sources, but how effective they will be with a vast, anarchic source such as the World Wide Web can only be speculated upon. The lack of literature found on distributed vector space models also suggests that many of the other techniques for information retrieval have not been explored in a distributed environment.

Section 4 outlines how a distributed collection can be organised. It first describes the possible application architectures and logical network topologies, and then it looks at some of the related issues in creating a distributed information retrieval system. These issues have included a look at how a simple distributed information retrieval system scales with varying sizes of workload. This study showed that scaling a distributed system is not a simple operation, but requires careful consideration of where bottlenecks will occur in the system. It has also shown that simulations of a system can provide an effective means for evaluating proposed system configurations in a controlled environment. Other issues considered in this section look at how to maintain the representation of a distributed collection over a distributed information retrieval system, as well as four techniques for merging results from collections. This section then describes several different distributed information retrieval systems. These systems only cover a fraction of the possible organisations that could be used for a distributed information retrieval system, and many of them have only had limited testing on relatively small, stable collections.

This literature survey has reviewed the many benefits that a distributed information retrieval system can offer for access to information sources. From the various studies that have been carried out, it is however evident that the possibilities offered by such systems have by no means been fully explored. Only a few of the various techniques and potential architectures for these systems are reviewed. How well many of these would cope with a large, dynamic information source such as the Internet is still speculation.

B.7 References

[1] BARJA, M. L., BRATVOLD, T., MYLLYMAKI, J. AND SONNENNBERGER, G. Informia: a mediator for integrated access to heterogeneous information sources. In *CIKM*, pages 234-241, 1998.

[2] BERRY, M.W. Introduction to Vector Space Models (http://www.cs.utk.edu/~berry /lsi++/node4.html). 1996.

[3] BOWMAN, C.M., DANZIG, P.B., HARDY, D.R., MANBER, U. AND SCHWARTZ, M.F. The Harvest information discovery and access system. In *Computer Networks and ISDN Systems*, 28 (1995), pages 119-125, 1995.

[4] BOWMAN, C.M., DANZIG, P.B., HARDY, D.R., MANBER, U., SCHWARTZ, M.F. AND WESSELS, D.P. Harvest: A scalable, customisable discovery and access system. University of Colorado, Boulder, Technical Report CU-CS-732-94, 1995.

[5] CAHOON, B. AND MCKINLEY, K.S. Performance evaluation of a distributed architecture for information retrieval. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Zurich, Switzerland, Aug. 1996), pages 110-118, 1996.

[6] CAHOON, B., MCKINLEY, K.S. AND LU, Z. Evaluating the performance of distributed architectures for information retrieval using a variety of workloads. To appear in *IEEE Transactions on Information Systems*.

[7] CALLAN, J.P., CROFT, W.B. AND HARDING, S.M. The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications* (Valencia, Spain, 1992), pages 78-83, 1992.

[8] CALLAN, J.P., LU, Z. AND CROFT, W.B. Searching distributed collections with inference networks. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA, July 1995), pages 21-28, 1995.

[9] JAIN, A. AND DUBES, R. *Algorithms for clustering data.* Prentice Hall, 1988.

[10] KANTOR, B. AND LAPSLEY, P. Network news transfer protocol. *Request for comments 977*, Network Working Group, 1986.

[11] KOSMYNIN, A. Distribution in Internet Resource Discovery (http://archive.dstc. edu.au/AW3TC/papers/kosmynin). 2000.

[12] LU, Z. AND MCKINLEY, K.S. Partial replica selection based on relevance for information retrieval. In *Proceedings of the Twenty-second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkley, California, USA, 1999), pages 97-104, 1999.

[13] LU, Z. AND MCKINLEY, K.S. The effect of collection organization and query locality on information retrieval system performance. In *Advances in Information Retrieval,* pages 73-202.

[14] OIKARINEN AND REED. Internet relay chat protocol. *Request for comments 1459,* Network Working Group, 1993.

[15] PONTE, J.M. Ph.D. Thesis, University of Massachusetts.

[16] PONTE, J.M. AND CROFT, W.B. A language modeling approach to information retrieval. In *Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* pages 275-281, 1998.

[17] SELBERG, E. AND ETZIONI, O. On the stability of web search engines. In *RIAO'2000,* pages 235-247, 2000.

[18] SHELDON, M.A., DUDA, A., WEISS, R. AND GIFFORD, D.K. Discover: a resource discovery system based on content routing. In *Proceedings of the Third International World Wide Web Conference* (Darmstadt, Germany, 1995), 1995.

[19] SIMON, E. *Distributed Information Systems.* London, McGraw-Hill, 1996.

[20] SONG, F. AND CROFT, W.B. A general language model for information retrieval. In *Proceedings of the Twenty-second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkley, California, USA, 1999), pages 279-280, 1999.

[21] VAN RIJSBERGEN, C.J. *Information Retrieval.* 2nd ed. Butterworth & Co, 1979.

[22] VILES, C. L. Maintaining state in a distributed retrieval system. In *32nd ACM Southeast Conference* (Tuscaloosa, Alabama, March 1994), pages 157-161, 1994.

[23] VILES, C.L. AND FRENCH, J.C. Dissemination of collection wide information in a distributed information retrieval system. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA, July 1995), pages 12-20, 1995.

[24] WEISS, R., VELEZ, B., SHELDON, M.A., NAMPREMPRE, C., SZILAGYI, P., DUDA, A. AND GIFFORD, D.K. HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *Proceedings of the 1996 Seventh ACM Conference on Hypertext* (Washington D.C., USA, March 1996), 1996.

[25] WITTEN, I.H., MOFFAT, A. AND BELL, T.C. *Managing Gigabytes.* 2nd ed., Morgan Kaufmann, 1999.

[26] XU, J. AND CROFT, W. B. Cluster based language models for distributed retrieval. In Proceedings of the Twenty-second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (Berkley, California, USA, 1999), pages 254-261, 1999.