ProbFuse: Probabilistic Data Fusion

by

David J. Lillis

A Thesis submitted to
University College Dublin
for the degree of Master of Science
in the
College of Engineering, Mathematical
and Physical Sciences

February 2006

School of Computer Science and Informatics

Professor Barry Smyth (Head of Department)

Under the supervision of

Dr. Rem Collier
Mr. John Dunnion

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In recent years, the proliferation of information being made available in such domains as the World Wide Web, corporate intranets and knowledge management systems and the "information overload" problem have caused Information Retrieval(IR) to change from a niche research area into a multi-billion dollar industry.

Many approaches to this task of identifying documents that satisfy a user's information need have been proposed by numerous researchers. Due to this diversity of methods employed to perform IR, retrieval systems rarely return the same documents in response to the same queries.

This has led to research being carried out in the fields of data fusion and metasearch, which seek to improve the quality of the results being presented to the user by combining the outputs of multiple IR algorithms or systems into a single result set.

This thesis introduces *probFuse*, a probabilistic data fusion algorithm. *ProbFuse* uses the results of a number of training queries to build a profile of the distribution of relevant documents in the result sets that are produced by its various input systems. These distributions are used to calculate the probability of relevance for documents returned in subsequent result sets and this is used to produce a final fused result set to be returned to the user.

ProbFuse has been evaluated on a number of test collections, ranging from small collections such as Cranfield and LISA to the Web Track collection from the TREC-2004 conference. For each of these collections, *probFuse* achieved significantly superior performance to CombMNZ, a data fusion algorithm often used as baseline against which to compare new techniques.

ACKNOWLEDGEMENTS

A special word of thanks is due to my supervisors. Dr. Rem Collier has been tremendous for discussing research ideas, for providing wonderful guidance on the direction of my research, and for working so hard in helping this thesis to be written.

John Dunnion's insight has been invaluable in making my research what it is. He has greatly enhanced my knowledge of IR and computer science in general.

An immense debt of gratitude is owed to Fergus Toolan, for being constantly available to provide guidance, for his help on so many things and for his fiery enthusiasm for my work.

Above all, Rem, John and Fergus have made my research an enjoyable experience and I am privileged to have worked with them.

My family also deserve special recognition. They have been a constant source of encouragement and support throughout my academic career and indeed throughout my life.

I owe a huge amount of thanks to Hailey, to the extent that thanking her properly would take up an inordinate amount of this thesis. Thanks so much for everything.

A number of other people also deserve to be thanked here: Eamonn Newman, who started me off with $\LaTeX$, without which this thesis, quite literally, wouldn't be the same; Keith Bradley, who provided a number of references to interesting papers on web search and provided valuable input to a number of ideas that were run by him; The members of the HOTAIR project group that contributed hugely to my research and from whom I learned a lot; The members of the IIRG, particularly those who subjected themselves to my ramblings on multiple occasions; Ellen Voorhees at NIST and the other TREC organisers and participants for providing the data for my experiments; and John Doody and Edwin Costello, who ensured that both my sanity and my ridiculous caffeine addiction were maintained over the past eighteen months.

LIST OF PUBLICATIONS

- David Lillis, Fergus Toolan, Angel Mur, Liu Peng, Rem Collier and John Dunnion. Probability-based fusion of information retrieval result sets. In *Proceedings of the 16th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2005)*, pages 147-156, 7-9 September, Portstewart, Northern Ireland, 2005, University of Ulster.
- Liu Peng, Rem Collier, Angel Mur, David Lillis, Fergus Toolan and John Dunnion, A Self-Configuring Agent-Based Document Indexing System, in *Proceedings of the 4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2005)*, pages 624-627, 15-17 September, Budapest, Hungary, 2005.
- Angel Mur, Liu Peng, Rem Collier, David Lillis, Fergus Toolan and John Dunnion, A HOTAIR Scalability Model, in *Proceedings of the 16th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2005)*, pages 359-368, 7-9 September, Portstewart, Northern Ireland, 2005.
- David Lillis, Fergus Toolan, Rem Collier and John Dunnion, ProbFuse: A Probabilistic Approach to Data Fusion, in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, Seattle, 2006.
- David Lillis, Fergus Toolan, Rem Collier and John Dunnion, Probabilistic Data Fusion on a Large Document Collection, in *Proceedings of the 17th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2006)*, 11-13 September, Belfast, Northern Ireland, 2006
- David Lillis, Fergus Toolan, Angel Mur, Liu Peng, Rem Collier and John Dunnion, Probability-Based Fusion of Information Retrieval Result Sets, accepted by *Artificial Intelligence Review, Special Edition on Artificial Intelligence and Cognitive Science (AICS-05)*, 2006.

CHAPTER

ONE

Introduction

Information Retrieval (IR) is a long-established research area in the short history of computing. The term "Information Retrieval" itself was coined by Mooers as early as the 1950s.

> Information retrieval is the name of the process or method whereby a prospective user of information is able to convert his need for information into an actual list of citations to documents in storage containing information useful to him [38].

The roots of IR lie in the organisation of libraries. As library collections increased, more sophisticated methods became necessary in order to aid library users in finding books that were relevant to their areas of interest. Typically, these were stored as card catalogues, which were maintained manually. These catalogues would allow a librarian to perform a search on a user's behalf by searching for information such as authors' names, book titles and keywords.

With the advent of computers, it was a logical step to maintain catalogues in electronic form that would provide faster access than the old paper-based systems. However, these computer systems are still best described as performing "Data Retrieval", rather than "Information Retrieval". This is because books were being searched for based on information about them, such as title or author, rather than the information contained in them.

In order for the contents of books and documents to be available for searching, it was necessary develop a method for computers to be able to store some searchable representation of the text contained in them. Progress on this task was made during the 1950s and it is proposed that the ICSI conference in Washington in 1958 marks the beginning of true Information Retrieval [28].

Initially research was performed on small collections of documents, due to limitations on the storage space and processing power available. For example, the well-known SMART project performed its early experiments on a "few hundred" document abstracts [45].

Improvements in computing technology have allowed modern IR systems to make huge amounts of textual information available to their users. This has coincided with the popularity of the World Wide Web, which has caused an explosion in the amount of information being published. In the modern age, publishing is no longer the sole preserve of government agencies and traditional publishers. Anybody can easily publish information on the Web.

Due to this huge increase in the amount of textual information available, IR systems have taken on a more important role than ever before. Users are bombarded with massive amounts of information that is impossible for the human brain to process. This phenomenon has become known as "Information Overload" [52] and also applies to other information sources such as corporate intranets and knowledge management systems.

IR systems have become a necessary tool for Internet users to find the information they require. The popular Google search engine searches billions of documents and its popularity is reflected by the word "googling" entering people's everyday vocabularies [12]. One study found that by the year 2000, 85% of Web users use search engines to find information [32]. This figure is likely to be much higher now and emphasises the importance of IR in modern society.

A typical IR system allows users to express an information need in the form of a query, which is used to identify documents that contain information relevant to the user's request, thus satisfying this need. Historically, this query would be expressed in terms of a Boolean query, which combines keywords with the *AND*, *OR* or *NOT* operators to indicate which combination of keywords should be present in or absent from documents retrieved. More recently, these queries can take the form of full text queries or simple lists of keywords for which to search [30].

The usual output of an IR system is a list of documents that it estimates to be relevant to the user's query. That is, that the document includes information that helps to satisfy the user's information need. This list is typically ranked according to how relevant to the query the IR system believes the documents to be, with those documents with the greatest degree of relevance being returned at the beginning. The ultimate goal of any IR system is outlined by Van

Rijsbergen [43]:

> The purpose of an automatic retrieval strategy is to retrieve all the *relevant* documents at the same time retrieving as few of the *non-relevant* as possible.

Numerous computational approaches have been proposed to address the challenges associated with IR. In addition to standalone IR systems, meta search engines such as ProFusion [19], SavvySearch [26] and MetaCrawler [49] have been developed to relay users' queries to multiple search engines and combining the results that they return.

1.1 Motivations for Data Fusion

No single approach to IR has been shown to achieve superior performance in all situations. This is because of the different methods of representing documents and user queries, the different policies regarding document and query preprocessing and the different algorithms used to rank documents. As a result, individual IR systems will retrieve different documents from the same document collection in response to the same query [15]. This was shown in the entries for the TREC-1 conference, many of which achieved approximately the same performance level but returned substantially different documents in their result sets [23].

It has been shown that retrieval performance can be improved by combining the lists of documents produced by a number of different IR algorithms into a single list [4]. This has become known as "data fusion" [1]. Because some IR algorithms will return documents that are missed by others, combining numerous sets of results in the correct way will lead to more relevant documents being presented to the user. In addition, it has been shown that the presence of a document in the results returned by a number of IR systems reflects an increased probability that it is relevant to the given query [48].

The term "data fusion" specifically refers to the task of using a number of different IR systems to retrieve documents from the same document collection. This is in contrast to related tasks such as "collection fusion", where the document collections being searched are distinct [57] and situations where there is only partial overlap between the document collections [64]. A successful data fusion algorithm should:

- retrieve more relevant documents by using many different IR systems as its inputs
- return relevant documents in high positions in the fused result set by examining the extent to which the input systems agree on the relevance of documents
- scale to large collections without degradation in performance
- be robust to differences in the completeness of relevance judgments if these are required for fusion to be performed
- ensure that the searches performed by the individual IR systems are executed in parallel, in order to ensure a prompt response to the user

1.2 Core Contributions

This thesis introduces *probFuse*, a novel probabilistic approach to the data fusion problem. *ProbFuse* is based on the intuitive assumption that the performance of individual IR systems on a training set of historic queries is indicative of future performance. By analysing the distribution of relevant documents in the training result sets, it is possible to estimate the probability that a document appearing in a particular position in a result set will be relevant to the given query.

This thesis describes a number of experiments that demonstrate the effectiveness of *probFuse* and show its improvement over the CombMNZ algorithm, which is commonly used as a baseline for new data fusion algorithms [6] [37]. *ProbFuse* is shown to achieve encouraging results on a number of document collections. The document collections used range from small collections such as the Cranfield and LISA collections to the large web track collection from the TREC-2004 conference.

Achieving performance gains for such a wide variety of document collections is an extremely encouraging result. This demonstrates that *probFuse* can scale to many sizes of document collection while continuing to maintain high performance.

1.3 Thesis Overview

Chapter 2 outlines a number of approaches taken by other researchers in the past to perform data fusion and related tasks. In addition, some basic IR techniques are introduced, along with some common measure used to evaluate the performance of IR systems.

The *probFuse* algorithm is introduced in Chapter 3. This relies on data from the past performance of its input systems to estimate the relevance of documents returned in response to subsequent queries. Two methods of calculating the probability that a document is relevant to a given query are proposed.

An exploratory study was initially carried out in order to test the effectiveness of *probFuse* on a number of small document collections, comparing it with the performance of the popular CombMNZ fusion algorithm. This study is described in Chapter 4. These small document collections include complete relevance judgments, meaning that the relevance of each document to each of the accompanying queries is known. This facilitates the calculation of the probability of relevance, which is necessary for *probFuse* to perform effective data fusion.

Chapter 5 describes experiments on larger document collections from two TREC conferences, using submissions for the ad hoc track of the TREC-3 and TREC-5 conferences as the inputs to be fused. In these experiments, the performance of two variations of *probFuse* that calculate their probabilities in different ways is examined. The aim of these experiments is to demonstrate that *probFuse* achieves superior performance to CombMNZ on these collections. Additionally, it is also necessary to empirically identify values for two variables that are central to the *probFuse* algorithm so as to achieve optimal fusion performance.

Following these experiments, the effectiveness of the values identified in Chapter 5 was tested by performing fusion on input result sets generated from a much larger test collection. This experiment is presented in Chapter 6. Additionally, the effect of increasing the level of incompleteness of the available relevance judgements is examined. This experiment uses the document collection from the web track of the TREC-2004 conference.

Finally, Chapter 7 presents the conclusions that can be drawn from the research presented in this thesis, along with ideas regarding directions for future related work.

CHAPTER

TWO

Background Reading

2.1 Introduction

Numerous researchers have proposed a range of different approaches to IR. Each of these approaches has a different method of representing a document in a database, along with an algorithm that it uses to compare users' queries to these documents in order to identify those that are most relevant to the query.

In addition to the different models being used, IR systems may also use various preprocessing steps for their queries and document indexes. These include stopword removal (the process of removing common words from a document collection, as they are of limited use for discriminating between relevant and nonrelevant documents) and stemming (a process of suffix-stripping so that words that are semantically linked will be treated as such in the index) [39]. Distinct IR systems may have contrasting concepts of what constitutes a stopword and may also use different stemming algorithms.

The primary focus of this chapter is to describe work done by previous researchers in the area of data fusion. Because data fusion requires underlying IR systems to produce result sets for fusion, a number of models that have been developed to perform IR are first introduced (Section 2.2). Section 2.3 outlines a number of measures that have been proposed to evaluate the performance of IR systems. Finally, in Section 2.4, a number of previous approaches that have been taken by other researchers in the past to perform data fusion and related tasks are discussed.

2.2 Basic Information Retrieval Models

Numerous IR models have been proposed to solve the problem of identifying documents in a collection that are relevant to a given query.

These IR techniques typically assign a score to each document in a collection. This score is a judgment of that document's relevance to the given query. A list of documents, ranked according to this relevance score, is then returned. These ranked lists are known by numerous names in the relevant literature. For consistency, they are referred to as *result sets* in this thesis, as in [6].

This section describes a number of models that have been proposed for IR. In Chapter 4, three of these are used to produce the inputs to an initial exploratory study. For each of these models, the Vector Space Model, the Fuzzy Set Model and the Extended Boolean Model, the formulae that used to implement the technique are included. In each case the implementation details are taken from [3].

2.2.1 Boolean Model

The Boolean Model of Information Retrieval is based on Boolean algebra and set theory. Under this model, a document in the collection is judged to be relevant or non-relevant to the given query by considering whether query terms appear in the document or not. Its principal advantage lies in its simplicity, in that for each relevance judgment, no information other than the query and the document itself is required. Its query language is also simple, consisting of keywords linked with the operators AND, OR and NOT. Its main drawback is that it cannot distinguish between levels of relevancy and fails to return partial matches, since its judgment is a binary one. As such, it is considered by Baeza-Yates and Ribeiro-Neto as more of a data retrieval model than an information retrieval model [3].

2.2.2 Vector Space Model

The Vector Space Model relies on a non-binary weighting system to rank documents in order of their relevance to a given query. Each term is assigned a weight in respect of each document. Each document is then represented by a t-dimensional vector of these weights, where t is the total number of terms present in the document collection. The similarity between a query and a document is then calculated as the cosine of the angle between their respective vectors [47] [44].

A key benefit of the Vector Space Model is that a document will not be automatically judged as being non-relevant as a result of not containing all the term in the query. A similarity threshold may be introduced to prevent documents with too low a similarity score being returned by the system.

The most popular method for calculating the weights for each term relies on term frequency (tf) and inverse document frequency (idf). The tf represents how often the term appears in the document in question. The reasoning behind this is that if a query term occurs frequently in a document, it is more likely to be relevant to the query. Because of the possibility of a document being judged relevant simply because it is a long document and contains many occurrences of a query term for this reason, it is necessary to normalise the tf score to remove such a possibility. The tf ($f_{i,j}$) of term i in document j is calculated by

$$f_{i,j} = \frac{freq_{i,j}}{max_l \, freq_{l,j}} \tag{2.1}$$

where $freq_{i,j}$ is the number of times term i appears in document j and $max_l \, freq_{l,j}$ is the maximum number of times any term l appears in document j. For any term i not appearing in document j, $f_{i,j} = 0$.

The idf factor is necessary to ensure that no query term has a disproportionate effect on the judgment of relevance. It is based on the proportion of the total documents in the collection that the term appears in. A term that appears in many documents in the collection is less likely to be a useful indicator of relevance than a term that only appears in a few documents. The idf score ensures that a greater weight is assigned to the term that occurs less frequently in the collection. The idf score of term i is given by

$$idf_i = log\frac{N}{n_i} \tag{2.2}$$

where N is the total number of documents in the document collection and n_i is the number of documents that contain a term i.

Once tf and idf have been calculated, $w_{i,j}$, the term weight for term i in document j is given by

$$w_{i,j} = f_{i,j} \times idf_i \tag{2.3}$$

Each document in the system is represented by a t-dimensional vector of these term weights, where t is the total number of terms that are present in the system. Each query submitted to the system is represented in the same way. The similarity between a document and a given query is then found by calculating the cosine of the angle between the two vectors. The similarity of document d_j to query q, $sim(d_j, q)$ is given by

$$sim(d_j, q) = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}} \tag{2.4}$$

where t is the total number of terms that are present in the system, $w_{i,j}$ is the term weight of term i for document j and $w_{i,q}$ is the term weight of term i for the query q.

The Vector Space Model has been used to great effect in empirical studies and has become very popular amongst the Information Retrieval community [23].

2.2.3 Probabilistic Model

The Probabilistic Model was originally proposed by Maron and Kuhns [35]. A Probabilistic IR system attempts to calculate the probability of each document being relevant to a particular query [27]. As with the Vector Space Model, relevance is not a binary attribute, so some documents may be judged to be more relevant than others.

For best performance, the model relies on information about relevant documents to estimate the relevance of other documents. In many systems (including a web-based search engine), no relevance information is available when the search is initiated. Thus tf and idf can be used to estimate relevance at this stage. Relevance information is frequently obtained by using user feedback to refine the search.

2.2.4 Fuzzy Set Model

Whereas the Boolean Model makes a binary judgment on whether a document belongs to the set of relevant documents or not, the Fuzzy Set Model is based on the theory that each document has a degree of membership of this set. Based on fuzzy set theory [67], it is necessary to define a membership function in order to determine the degree to which a document is part of a set. Once this is done, the union and intersections of fuzzy sets can also be calculated.

Within this model is the ability to create a thesaurus of similar terms (called a keyword connection matrix). This facilitates the creation of a fuzzy set for each term consisting of the documents which have a degree of relevance to that term. Unlike the more traditional IR approaches, this will not just consist of documents that contain the term itself but also documents containing similar terms. Once a query is run, the results are calculated by finding the intersection of the fuzzy sets relating to the query terms.

To construct this keyword connection matrix, it is necessary to calculate the correlation between each pair of terms in the document collection. The correlation $c_{i,l}$ between two terms k_i and k_l is given by

$$c_{i,l} = \frac{n_{i,l}}{n_i + n_l - n_{i,l}} \tag{2.5}$$

where n_i is the number of documents that contains the term k_i, n_l is the number of documents containing the term k_l and $n_{i,l}$ is the number of documents containing both term k_i and k_l.

Once this connection matrix has been constructed, a fuzzy set associated with each term can be defined. Each document d_j in the collection will have a degree of membership $\quad_{i,j}$ of the set for each term k_i. This is given by

$$\quad_{i,j} = 1 - \prod_{k_l \in d_j} (1 - c_{i,l}) \tag{2.6}$$

where k_l is a term contained in document d_j and $c_{i,l}$ is the correlation between term k_i and k_l.

The Fuzzy Set Model expects the queries it receives to be in Boolean form. In Chapter 4, Fuzzy Set Model is used as one of the inputs for the initial exploratory study in data fusion. The queries for each of the document collections used are in full text form. These are converted to Boolean type queries by inserting the AND operator between each pair of terms. In this situation, the degree of membership $\quad q, j$ of a document d_j to the fuzzy set D_q representing a query q can be simplified to

$${}_{q,j} = \prod_{i=1}^{t} {}_{i,j} \tag{2.7}$$

where t is the number of terms in the conjunctive Boolean query and ${}_{i,j}$ is the degree of membership of document d_j to the fuzzy set associated with term k_i.

A full discussion of the treatment of more complex Boolean queries is outside the scope of this thesis. Such a discussion can be found in [3].

The key advantage of the Fuzzy Set Model is the ability to take terms that are relevant to the query terms but are not included in the query themselves. However, documents containing all search terms will be given a score of 1, as with the Boolean Model. For this reason, the Fuzzy Set Model does not place any particular order on these documents and so retrieval performance suffers [22]. Thus the Fuzzy Set Model is not popular amongst IR researchers.

2.2.5 Extended Boolean Model

The Extended Boolean Model extends the simple Boolean Model to allow partial matching and term weighting by incorporating elements of the Vector Space Model, namely tf and idf [46]. The reasoning behind its development is the binary nature of the Boolean Model. For example, in a Boolean query requiring the presence of two query terms, a document will not be returned if only one of those terms appears, even though it is clearly more relevant than one in which none of the terms occurs.

The weights used in the Extended Boolean Model are slightly different to those used in the Vector Space Model. Instead of using the $tf.idf$ weighting scheme shown in equation 2.3, a normalised variation is used, which ensures that the weights lie between 0 and 1. This is given by

$$w_{x,j} = f_{x,j} \times \frac{idf_x}{max_i \; idf_i} \tag{2.8}$$

where $f_{x,j}$ is the term frequency of term x in document j as calculated by equation 2.1, idf_x is the inverse document frequency of term x, as calculated by equation 2.2 and $max_i \; idf_i$ is the maximum idf score for any term i in the document collection.

A version of the Extended Boolean Model, the p-norm model, introduces a variable p, which affects the way the model performs. If p is assigned a value of 1, the model functions similarly to the Vector Space Model. If p is assigned a value of infinity, it functions like the Fuzzy Set Model. The ranking behaviour of the model can be varied by using different values of p.

The p-norm variation of the Extended Boolean Model is used in Chapter 4. For each of the document collections used, the available queries are in the form of full text queries. In order to use the Extended Boolean Model, it was necessary to convert these to Boolean queries. To do this, it was assumed that the query was intending to find documents that contained all of the terms contained in it and a conjunctive Boolean query was formed by using the AND operator. The similarity between q_{and}, a conjunctive Boolean query with m terms and a document d_j, $sim(q_{and}, d_j)$ is given by

$$sim(q_{and}, d_j) = 1 - \left(\frac{(1-x_1)^p + (1-x_2)^p + ... + (1-x_m)^p}{m} \right) \tag{2.9}$$

where each x_i represents the weight $w_{i,j}$ associated with term k_i and document d_j.

This is only one form of query that the Extended Boolean Model can accept. Further details on running disjunctive queries and combining conjunctive and disjunctive queries can be found in [3].

Although introduced some time ago, it has not proved to be particularly popular amongst the IR community.

2.2.6 Neural Network Model

The Neural Network Model involves an IR system comprised of a three layer neural network [62]. The first layer contains nodes relating to the query terms, the second contains nodes relating to the terms in the collection and the third relates to the documents in the collection. When a query is run, the query term nodes send signals to the term nodes, which in turn send signals to the documents nodes, based on the weight of each term within each document. Once these signals reach the document nodes, they in turn send out signals to each term which has a weight in relation to that document associated with it. Because each node sends signals to multiple other nodes, these signals are weakened with each iteration. The document and term nodes continue to pass signals back and forth until these signals become extremely small.

This activation spread is considered to be analogous to an automated inbuilt thesaurus being invoked with each query (thus allowing relevant terms which do not themselves occur in the query to be taken into account), which is a strong argument in its favour [3].

2.3 Evaluation Measures

One of the key challenges of IR is to be able to evaluate the effectiveness of an IR system. The "Cranfield paradigm" for IR evaluation emerged in the 1960s. This stemmed from research carried out at the Cranfield College of Aeronautics to test indexing techniques. In their experiments, a number of test queries were created for a static document collection, and each document in the collection was judged to be either relevant or nonrelevant to each of the queries [41]. Despite the fact that the concept of "relevance" is quite subjective, it is the pillar upon which IR evaluation measures are based.

This section introduces a number of measures that are used in IR evaluation. It pays particular attention to those evaluation measures are will used in this thesis.

2.3.1 Precision and Recall

Precision and Recall are evaluation measures that reflect the key aims of any IR system, as outlined in Chapter 1. Recall measures a system's success at returning the maximum number of relevant documents possible. Precision measures the ability to avoid returning nonrelevant documents.

More formally, Recall is the fraction of the total available relevant documents that have been retrieved. It is given by

$$Recall = \frac{|R_a|}{|R|} \tag{2.10}$$

where $|R_a|$ is the number of relevant documents that have been retrieved and $|R|$ is the total number of relevant documents that are contained in the document collection.

Precision is the fraction of the total number of documents that have been retrieved that are relevant. It is given by

$$Precision = \frac{|R_a|}{|A|} \tag{2.11}$$

where $|R_a|$ is the number of relevant documents that have been retrieved and $|A|$ is the number of documents in the result set that is returned.

Precision and recall are somewhat contradictory in nature. Full recall can be achieved by simply including every document in the collection in the returned result set. However precision would be extremely low in this situation. Similarly, precision will be high in situations where only a few documents that are considered to be highly relevant to the query are returned.

A number of evaluation measures have been proposed that are based on recall and precision but attempt to allow for this trade-off.

2.3.2 Interpolated Precision and Recall

By themselves, recall and precision do not take the positions in which relevant documents are returned into account. Because result sets take the form of ranked lists, it is preferable for relevant documents to be returned at the beginning of this list. As a user proceeds down a result set, the precision and recall values vary. Plotting an interpolated precision versus recall graph demonstrates this effect. To plot this graph, precision is calculated for 11 standard recall levels, which are 0%, 10%, 20%, ..., 100%.

However, the recall levels for each query may not correspond with these standard recall levels. For example, if there are 3 judged relevant documents for the given query, retrieving the first will cause recall to rise from 0% to 33.33%. To cater for this, an interpolation procedure is necessary. The interpolated precision at each standard recall level is the maximum precision found between between it and the next largest recall level.

Interpolated precision/recall graphs allow both precision and recall to be viewed in a single graph. The performance of the retrieval system at different points in the result set can then be evaluated.

2.3.3 Mean Average Precision

Mean Average Precision (MAP) is defined as the mean of the precision scores obtained after each relevant document is retrieved, using zero as the precision for relevant documents that are not retrieved [8].

This measure rewards systems that return relevant documents in early positions in their result sets, as the precision at these points will be high. Although MAP does not specifically take recall into account, it does implicitly reward systems that achieve high recall. Because relevant documents that are not returned are assigned a precision value of zero, this punishes low-recall systems that return fewer of the relevant documents.

2.3.4 bpref

The bpref evaluation measure was proposed by Buckley and Voorhees to cater for situations where incomplete relevance judgments were available [8]. Relevance judgments are considered to be incomplete when not all documents have been judged for every test query. This means that documents fall into three categories rather than two: judged relevant, judged nonrelevant and unjudged.

Older evaluation measures assume that unjudged documents are nonrelevant. In contrast, bpref only takes judged documents into account, ignoring any documents for which a relevance judgement is unavailable. For a query with R relevant documents, the bpref score is given by

$$bpref = \frac{1}{R}\sum_{r} 1 - \frac{|n\ ranked\ higher\ than\ r|}{R} \tag{2.12}$$

where r is a relevant document and n is one of the first R nonrelevant document retrieved by the system.

2.3.5 Other Evaluation Measures

Other evaluation measures have also been proposed that aim to give a single value to measure IR effectiveness.

R-precision measures the precision after R documents have been returned, where R is the number of judged relevant documents that are available to be retrieved. 100% R-precision would mean that all the relevant documents were returned at the top of the result set.

P10 measures the precision after the first ten documents have been returned. This measure rewards IR systems with a tendency to return relevant documents in early positions. This is in contrast to the MAP measure, which measure performance across the full document ranking, including documents that a user is unlikely to ever examine.

The Harmonic Mean is a single-value measure that aims to reward IR systems that achieve a balance between precision and recall. The *E* Measure is a variation on the Harmonic Mean that allows users to specify whether recall or precision should be considered to be more important. It incorporates an adjustable parameter that causes the measure to favour either precision or recall.

2.4 Data Fusion

In recent years, much research has been conducted into what has become known as *data fusion* [1], *collection fusion* [57] or *results merging* [64]. In this section, the differences between these terms are discussed and a number of approaches that have been proposed by other researchers in the past are outlined. As data fusion is the core focus of this thesis, most emphasis is placed on data fusion techniques. However, for completeness, research on related concepts is also included.

Section 2.4.1 discusses a number of factors that affect the choice of data fusion or related technique that is best suited to a particular situation. This enables a distinction to be made between data fusion and other similar concepts that are applicable in different situations. In Section 2.4.2 the steps involved in performing data fusion are discussed. Finally, Section 2.4.3 outlines a number of fusion techniques that have been developed in the past by other researchers. Of these, the CombMNZ algorithm has become the standard data fusion technique used as a baseline against which to evaluate new techniques [2] [6] [37] [63]. Throughout this thesis, it will be used as the baseline when evaluating the *probFuse* algorithm. For this reason, a detailed description of CombMNZ is included in Section 2.4.3.1.

2.4.1 Factors Influencing Fusion

2.4.1.1 Input Systems

The concept of an input system varies from technique to technique. Here, the "input system" is used as a general term to describe anything that returns a result set. This may range from autonomous search engines searching their own separate databases to multiple IR techniques searching in the same database. The three most common types of input system addressed in previous research are as follows:

1. *Metasearch* - This involves fusion of result sets returned by autonomous, complete search engines. Because these search engines are designed to act in a standalone fashion, rather than being specifically designed for use by a meta search engine, relevance scores are not necessarily made available to the meta search algorithm. Similarly, information about the contents of the database each engine has access to is typically unavailable (the Source-Metadata problem). Montague and Aslam describe this type of fusion as *External Metasearch* [37], to distinguish it from *Internal Metasearch*, below.

2. *Distributed Information Retrieval* - This describes a situation when numerous search engines are designed to co-operate within the same system. Each engine searches its own separate database and returns result sets which are then fused. Unlike metasearch, the separate engines provide output that is intended for use by a fusion algorithm, with judged relevance scores included. Information about the database of each engine is also typically available. Distributed IR usually involves a "broker" sending queries to individual servers and fusing the results.

3. *Internal Metasearch* - Here, numerous algorithms perform searches on the same database and a fusion algorithm is applied to their results in order to improve overall performance. As with Distributed IR, the relevance score associated with each document will be available to the fusion algorithm. The significant difference from Distributed IR is that information about the database being searched by each IR technique will not be useful in differentiating between them for fusion purposes.

2.4.1.2 Database Overlap

Another significant difference between the various fusion algorithms is the extent to which the databases being searched by each input system is expected to overlap. The level of overlap in the databases being searched will influence a fusion technique's treatment of duplicate documents being returned in multiple result sets. Three levels of overlap may possibly occur, depending on the type of fusion that is being carried out.

1. *Disjoint databases* - Fusion of result sets returned by input systems searching separate, autonomous document collections has become known as *collection fusion* [56].

2. *Identical databases* - Fusion of result sets returned by input systems searching identical databases has become known as *data fusion* [64].

3. *Overlap exists but the databases are not identical* - Wu and Crestani [64] note that little research has been done in this area, with most research concentrating on either identical or disjoint databases.

The different treatment of these three situations is outlined in [64]. For disjoint or nearly disjoint databases, the possibility of numerous systems returning duplicate documents is remote and so can be safely ignored. Because the presence of duplicate documents is so unlikely, it would not be useful to use such duplication as increased evidence of relevance.

In contrast to this, when identical databases are being searched, the presence of a document in multiple result sets is usually assumed to infer relevance.

The treatment of duplicates in the third situation is more difficult. In this case, when a document appears in one result set but fails to appear in another, it can be explained by one of two reasons:

1. The document is contained in both databases but is not considered to be relevant to the query by one system.

2. The document is only contained in one database and so can only possibly be returned by one system.

Without having access to the entire contents of both databases, it is impossible to tell which of these situations has occurred.

Data fusion is useful in the context of an IR system that uses multiple techniques to retrieve relevant documents. For example, in the context of web page retrieval, one technique may use a full-text query on the textual content of documents, another may make use of the text associated with hyperlinks to or from a document and another may perform retrieval based on meta information associated with each document. As these individual retrieval techniques would be operating within a cooperative system, each will perform retrieval from the same document collection. In such a situation, data fusion facilitates the integration of the results produced by each of these distinct approaches in order to produce the output result set.

Collection fusion is the technique used by meta search engines such as ProFusion [19], SavvySearch [26] and MetaCrawler [49]. The input systems used in this situation are uncooperative and will retrieve documents from their own document collections.

2.4.2 The Fusion Process

There are a number of steps to be carried out in performing fusion. A fusion technique may incorporate any or all of Server Selection, Querying, Score Normalisation and Merging. These are discussed in the following sections.

2.4.2.1 Server Selection

The aim of Server Selection is to find a subset of the available systems to use for a given query. Most selection algorithms make their selection based on information they collect about the contents of the database each server searches. Thus it is mostly used in External Metasearch and Distributed IR. Selecting all available input systems is a valid server selection strategy, but is unnecessarily resource-intensive if it is possible to achieve comparable effectiveness using a subset of them [42].

Callan et al. [9] considered server selection to be analogous to document retrieval. Their CORI (Collection Retrieval Inference Network) used a variation of *tf.idf* document ranking to identify which collections should be used in processing the query. Instead of *tf* and *idf*, they made use of document frequency (the number of documents in the collection which contain the term) and inverse collection frequency (the number of collections which contain the term).

Using this approach, the authors recorded a reduction in recall. This is to be expected as the elimination of any collection which contains any relevant document will logically reduce the total number of relevant documents which it is possible for the system to return. The key finding of this research, however, was that precision at low recall levels was not significantly reduced when the number of collections used was reduced by as much as 43%. The difference in precision only became significant after 200 documents had been retrieved. In a typical Internet search engine, precision at low recall levels is the most important evaluation measure. This is shown in a study that found that 85% of web users looked at 10 documents at most when using search engines [51]. The authors suggest that if a user is interested in examining more than 200 documents, more collections could be searched.

An alternative method, *Lightweight Probes*, was used in [25] to select which servers will ultimately run the input query. On receipt of a query, the broker sends a probe to each available server. This probe consists of the two most significant terms in the query. The significance of the terms are calculated by reference to a Document Frequency Reference Collection, which is accessible to each broker. This reference collection should be capable of giving an approximation of the term frequency distribution of the real data in the distributed collections. Thus, it is unnecessary for this reference collection to grow proportionally with the actual data.

In response to this probe, each server returns information regarding the number of documents in which each of the terms appears, the number that contains both terms and the number that contains both terms in close proximity to each other. The broker can then use this data to decide which servers should be asked to process the entire query.

The amount of processing required of each server to satisfy the information requirement of the probe is much less than that needed to process an entire query, as no document ranking needs to be carried out. Also, the network bandwidth necessary is very low, as the amount of information travelling in each direction is small.

In experiments, it was demonstrated that this Lightweight Probes method outperformed the other methods it was compared to. *Server General Utility* assumes that some servers are superior sources of relevant information than others. However, it relies on data about past query processing on each server. Additionally, the authors noted that the server rankings are effectively fixed. Therefore, it is likely to be unsuited to systems where new servers may be added or removed on a continuous basis. *Collection Promise* assumes that sub-collections relate to particular categories of documents. This was the worst-performing approach tested, to the extent that the authors do not consider it worthwhile to carry out any further investigation on it.

In the gGlOSS system [21], the broker needs a small amount of information about each server. Firstly, it needs to know how many documents on the server each term appears in (this is the same as the document frequency measure in

[9]). Secondly, it needs the sum of the weight of each term over all documents on the server. It is estimated that the storage requirements for this information is 2% of that required to store a full combined index for all servers. Gravano et al compare a number of formulae to calculate an estimate of the ′goodness′ of each server. A threshold value (user-specified in this case) can be used to avoid searching judged non-useful servers.

2.4.2.2 Score Normalisation

For Distributed IR systems and systems where multiple algorithms are operating on the same database, each document returned in a result set will be accompanied by its associated relevance score. This may also be available for Metasearch. This score represents the underlying technique's estimation of the relevance of that document to the given query and each individual result set is ranked according to the assigned scores.

When fusing those result sets, however, the relevance scores emanating from various input systems are unlikely to be directly comparable. This is the case for two reasons.

When different IR techniques are being used to produce the result sets, they will have different ways of measuring relevance and will likely return their scores in different ranges. Table 2.1 illustrates this by showing the highest and lowest scores returned by six different systems in the ad-hoc task of the 3rd Text REtrieval Conference (TREC-3) [1].

Table 2.1: Highest and lowest raw scores returned by six IR systems in TREC-3

System	brkly6	eth001	nyuir1	pircs1	vtc5s2	westp1
Lowest	0.108848	0.06490	1902	0.7798	0.551542	0.633465
Highest	0.547715	0.63619	67848	9.8907	2.486964	0.828830

Using the raw scores, the scores returned by *nyuir1* will have a disproportionately large effect on the final order of the fused result set. Thus, it is necessary to make scores comparable between input systems. The process of scaling raw scores into comparable ranges is known as *normalisation*.

Due to what Larkey et. al. call the "rare term problem" [29], even scores generated by the same retrieval technique are not directly comparable when using different databases. This is because collection-dependent information is used in calculating relevance. The example given in [9] is that the idf scores for the

[1]The TREC conferences are introduced in more detail in Section 5.1

words “computer”, “tort” and “cholesterol” will vary widely among technical, legal and medical collections. They may be rare in other collections and so will be artificially rewarded in those collections. In this scenario, they suggest that a normalised idf could be calculated using idf information from the individual collections. This is computationally expensive, so they suggest the weighted scores approach instead, which shows similar results but is far less expensive. Each raw score is weighted according to a weight assigned to the collection it came from.

A normalisation scheme used by Fox and Shaw [18] is to manipulate each result set into a the range [0, 1]. The normalised score of each document in each result set is given by

$$normalised\_sim = \frac{unnormalised\_sim - min\_sim}{max\_sim - min\_sim} \tag{2.13}$$

where *max_sim* and *min_sim* are the maximum and minimum score, respectively, that are actually seen in the retrieval result.

Montague and Aslam describe this as “standard” normalisation [36] and argue that it is overly sensitive to outliers in the input result sets. This is because an unusually high maximum score in a result set would cause all other scores in that set to have a lower normalised score than they otherwise would. The opposite would be the case for an unusually low minimum score. In order to reduce outlier sensitivity, they proposed two alternative normalisation techniques.

Sum Norm, like standard normalisation, shifts the minimum score in the result set to 0. It then scales the set so that the sum of the normalised scores is 1. *ZMUV Norm* (Zero-Mean, Unit-Variance) shifts the mean of the scores in the result set to 0 and then scales the variance to 1. Both of these are less susceptible to outliers than standard normalisation, since the value used for scaling isn’t dependent on a single document relevance score.

One difference between ZMUV and the others is the treatment of unretrieved documents. With standard normalisation and Sum Norm, unretrieved documents are assigned a normalised score of zero. However, doing this for ZMUV would result in unretrieved documents having an average relevance score. Thus unretrieved documents are assigned a normalised score of -2 (i.e. two standard deviations below the mean). Noting that some common fusion algorithms assume that relevance scores are all greater than zero, the authors also proposed *2ZMUV*, which shifts the mean to 2, causing "most scores" to be positive and allowing them to be used in any fusion technique.

A slight variation of standard normalisation was used in [40] and [29]. Here, instead of using the maximum and minimum scores actually contained in the result set, they used the maximum score an ideal document could get for the given query in the input system in question and the corresponding minimum. Again, this is less susceptible to outliers than standard normalisation.

A very different normalisation technique was used by Wu and Crestani in their Shadow Document fusion methods [64]. They assume that there is a linear relationship between the scores assigned to the same document by different input systems. Regression analysis is then applied to pairs of result sets, beginning with those that have the greatest overlap.

A normalisation technique that is specific to disjoint databases was proposed by Si and Callan in [50]. Query-based sampling is used to get information about the contents of the databases. No co-operation is required from the input systems other than answering queries. Queries are sent to each source and the documents that are returned are analysed. These are then saved in a centralised database, which is a small subset of the entire distributed collection. This query-based sampling is based on [10].

At query-time the queries, in addition to being run by each individual input system, are also run over the small centralised collection. The normalisation technique is based on two hypotheses:

- Some documents returned by the databases will also appear in the centralised database. This is assumed because the databases to be queried are selected based on their resource description, which is built based on the centralised database.
- The scores assigned to these common documents by the individual input systems can be mapped to their centralised score using a regression function. A separate regression function can be used for each database and once this is known, it can be used to produce normalised scores for documents that do not appear in the centralised database.

In Distributed IR systems, an alternative to normalising scores is to use Collection Wide Information (CWI) to ensure that the scores returned by the input systems are directly comparable without normalisation.

De Kretser et al [16] took the approach of aggregating CWI at the broker before query time, which was then communicated to the input systems whenever a query was run. The focus of the paper was to determine how much knowledge a broker should have of the global environment and how much should be distributed amongst the input systems.

The *Central Nothing* approach required least information to be maintained. The broker maintained no CWI, only a list of input systems that were available to them. Each system evaluated queries independently within its own sub-collection. This kept global information to a minimum but at the significant cost of sacrificing much of the effectiveness of most ranking algorithms as the "rare term problem" becomes an issue.

Going to the other extreme, under the *Central Index* approach, the broker has access to the entire index of the collection. With this model, all query processing is carried out by the broker and the input systems' only function is to fetch the documents the broker requests from them. In this case, it was noted that the storage requirements at each broker were large, with the potential to become impractically so [21]. In addition, due to the fact that this central index must be processed sequentially, processing time was greater than with other approaches, with no noticeable gain in effectiveness. It may be argued that, in this model, the servers merely represent a storage area network, and not a true distributed IR system. It was noted in [40] that using global idf scores, the system would behave the same as a centralised system.

A superior approach to either of these is a *Central Vocabulary*, under which only the vocabulary of each sub-collection, rather than the entire index is maintained by the broker. Under such a scheme, each input system receives the query terms accompanied by the weight to be used for each. This allows ranking algorithms to be applied effectively and reduces the central storage requirements from that of Central Index.

Another approach, proposed by Viles and French [53], relies on servers sharing CWI amongst themselves. They performed experiments whereby each server had perfect knowledge of the documents it had control over, along with a portion of the information maintained by all other servers. Based on the premise that idf is merely an indicator of the significance of a term [54], rather than a strict mathematical value, the authors results demonstrate that it is possible to maintain retrieval effectiveness without each server having perfect knowledge of every other server.

Using small collections, they found that if each server had information about only 20% of each other server's CWI, retrieval effectiveness approached that of a centralised system. This value is based on random allocation of documents (i.e. each document has an equal probability of being stored on a particular server). Where document clustering was used to group similar documents together, the amount of CWI each server required to maintain effectiveness increased. The collections in question were Cranfield (1400 documents), CACM (3204 documents), MED (1033 documents) and CISI (1460 documents).

The benefits of such an approach are clear. There is no reliance on any single element in the system to maintain global collection information. This information can be periodically updated in the background, meaning that at query time, each server has access to all the information it needs to run the query, rather than requiring it to be supplied by the broker that calls it.

Early experiments showed distributed IR to have inferior performance to a centralised system [53][65]. However, Powell et. al. [40] argue that this is because all available databases were queried and describe experiments that show that appropriate collection selection can cause Distributed IR systems to have superior performance to a centralised system. Xu and Croft [66] were able to achieve performance similar to a centralised system by clustering documents into similar subjects and using collection selection to choose which clusters to query.

2.4.3 Fusion Techniques

There are two broad categories of fusion techniques. The first utilises the relevance scores returned by the individual techniques to produce an overall score on which documents are then ranked. Such techniques are useful in situations where the result sets to be fused are produced by multiple techniques within the one Information Retrieval system. However, with some meta search engines (which combine the results of distinct search engines), the relevance scores are not available, so the rank of each document in each result set is used to produce a combined relevance score.

Vogt and Cottrell [55] outline three effects that fusion techniques attempt to benefit from. If the individual systems are retrieving different documents, this is likely to increase recall (the fraction of total relevant documents that have been retrieved). They describe this as the "Skimming Effect", as a fusion technique would "skim" the top-ranked documents from each result set, since the highest density of relevant documents is most likely to appear there. They also describe the "Chorus Effect", in which several retrieval sources are in agreement that a document is relevant. In situations where this agreement is correct, fusion techniques which attach a greater significance to documents which are common to multiple sources will perform well.

They also identify a "Dark Horse Effect", in which one retrieval approach returns results of a much different quality than the others. This may either be the returning of unusually accurate or inaccurate results. Vogt and Cottrell note that the Chorus and Dark Horse effects are somewhat contradictory in nature, with the former encouraging fusion techniques to take as many input systems into account when fusing and the latter suggesting that a single input may provide the best performance.

2.4.3.1 Score-based Techniques

This section describes a number of fusion techniques that make use of the relevance score assigned to each document by each input system to calculate a score by which the fused result set will be ordered.

Linear Combination

The Linear Combination model involves a weight being calculated for each input system. That weight is multiplied by each document score the system returns, with the final fused score for a document being the sum of these weighted scores.

The model is formalised in [55] as follows:

$$(W, x, q) = \sum_{i=1}^{s} w_i \quad {}_i(x, q) \tag{2.14}$$

where s is the number of input systems, ${}_i(x, q)$ is the raw score given to document x by input system i in response to query q, w_i is the weight for input system i and (W, x, q) is the score used for including document x in a fused result set in response to query q given the set of weights W.

A Linear Combination was used by Callan et. al in [9], where the weight of each input system was a function of the database selection score as calculated by CORI (see Section 2.4.2.1. It was found that this produced similar results to using global *idf* scores, while being much less computationally expensive. This was also used in [40] and a variation using normalised scores (see Section

2.4.2.2) was used in [50] and [29].

A different weighting system was proposed in [42]. Named LMS (using result Length to calculate Merging Score), it relied on the number of documents returned by each input system. This was based on the hypothesis that systems returning more documents are more likely to be providing better results. Under LMS, the weight given to each input system is the number of documents returned by it, relative to the number returned by the other systems. Its principal advantage lies in its simplicity since no prior knowledge of the input systems is necessary. Disregarding server selection, Rasolofo et. al. found that LMS produced superior results to Callan's CORI method.

In [63], Wu and Crestani use the *WSUM* (Weighted SUM) technique to calculate three possible weights to be assigned to input systems: 'good', 'fair' and 'poor'. Which category each fits into is calculated by testing how much agreement there is between systems. For each system, they take the top N documents in its result set and sum the number of occurrences of these documents in all other result sets. The categorisation of a system as 'good', 'fair' or 'poor' is based on how this score compares with the average for all systems. The score used for each document for fusion was a linear combination of the document's normalised score (using standard normalisation) and the weights of the appropriate input systems. A variation of this was also proposed for situations where scores were not available. Here, a score was calculated based on the ranking of the document in the result sets and the linear combination was performed based on that.

The ProFusion meta search engine [19] makes use of a fusion technique similar to Linear Combination . Each input system is assigned a "confidence factor", which is based on its performance over 25 queries. Documents are given a "matching factor", which is their normalised score (the normalisation step is to divide each raw score by the maximum score attributed to any document in the result set). The confidence and matching factors are then multiplied to give the final ranking score for each document. If a document has been returned by a number of input systems, it is the maximum of these scores that is used in the fused result set, rather than the sum, which would be the case for a Linear Combination.

Comb* Algorithms

In [18], Fox and Shaw proposed a number of fusion algorithms based on normalised scores, which are outlined in Table 2.2. In that table, the "Relevance

Score for Fusion" is the score assigned to each document on which the fused result set will be ordered. The normalisation scheme used was standard normalisation, although other normalisation schemes have been used with these algorithms too. Montague and Aslam used Sum Norm and 2ZMUV in [36]. Their ZMUV normalisation scheme was unsuited to CombMNZ, since below-average scores were normalised to be negative. Lee also performed experiments with a rank-based variation of CombMNZ (see Section 2.4.3.2).

Table 2.2: Fox and Shaw's Comb* Algorithms

Name	Relevance Score for Fusion
CombMIN	Minimum of individual relevance scores
CombMED	Median of individual relevance scores
CombMAX	Maximum of individual relevance scores
CombSUM	Sum of individual relevance scores
CombANZ	CombSUM $\div$ number of non-zero relevance scores
CombMNZ	CombSUM $\times$ number of non-zero relevance scores

The fusion technique used by the *MetaCrawler* meta search engine [49] is the same as CombSUM, with the normalisation scheme being similar to standard normalisation except that scores are scaled in the range of [0, 1000]. SavvySearch [26] uses CombSUM with standard normalisation. Documents returned by input systems that do not return a relevance score are arbitrarily assigned a score of 0.5.

In order to run CombMNZ, two steps must be performed. Firstly, the scores attributed to each document by each input must be normalised, so that they lie in a common range. The normalisation scheme chosen for the experiments outlined in this thesis is the "Standard Normalisation" that is discussed in Section 2.4.2.2.

Once the scores have been normalised, $CombMNZ_d$, the CombMNZ ranking score for any document d is given by

$$CombMNZ_d = \sum_{s=1}^{S} N_{s,d} \times |N_d > 0| \tag{2.15}$$

where S is the number of result sets to be fused, $N_{s,d}$ is the normalised score of document d in result set s and $|N_d > 0|$ is the number of non-zero normalised scores given to d by any result set.

A number of experiments were performed by Lee [33] to compare the performance of these algorithms and he also attempted to understand situations in which fusion could produce superior performance to single systems. These experiments involved fusion of result sets from six randomly-selected systems from TREC-3. He found that CombMNZ performed best, followed by CombSUM. Because both of these algorithms use a document's presence in multiple result sets as evidence of relevance, they exploit the "Chorus Effect" and are best suited to data fusion.

Citing Belkin et. al. [7], Lee claimed that different query representations or retrieval techniques would result in significantly different sets of documents being retrieved. Based on this, he hypothesised that fusing two result sets would be beneficial if they had a greater overlap of relevant documents than of non-relevant documents. However, he did not attempt to specify an optimum ratio of overlap where fusion will be most effective.

This overlap hypothesis has been rejected by Beitzel et. al. [5][6] (this stemmed from research done by Chowdhury et. al. in [11]. In their experiments, they could find no correlation between differing levels of overlap and retrieval performance. They argue that in a single system using highly effective retrieval techniques, fusion will not improve performance above that of the best-performing individual technique.

They argue that fusing result sets from autonomous systems (as Lee did) varies more than just the retrieval strategy. Systemic differences such as different stopword lists, parsers, stemming algorithms and relevance feedback will also be introduced and it is this that causes the different systems to return different result sets. Lee did not take such factors into consideration and this, they argue, makes it difficult to isolate factors that contribute to fusion effectiveness. In addition, Lee did not select the best performing TREC systems for his experiments, selecting them randomly.

The work of Beitzel et. al. is based on two hypotheses:

1. Highly-effective IR systems can be assumed to return high rankings for relevant documents. This means that common, non-relevant documents are more likely to be boosted than common relevant documents, thus fusion actually harms retrieval effectiveness.

2. In systems where systemic variables are eliminated, highly-effective IR algorithms will return very similar result sets, with the ranking being the major difference. Therefore, it's unlikely that unique relevant documents will be merged into the final result set.

Their experiments were based on data from the Ad-hoc track of TREC 6,7 and 8, and the Web Track collections from TREC 9 and 10. Systemic properties (parsers, stemmers, phrase lists and stopword lists) were kept constant, so the only variable parameter was the retrieval strategy. Three retrieval techniques were used and the result sets fused using CombMNZ.

To compare with their system, they also used CombMNZ to fuse the result sets of the three best systems from the same TREC conferences. For the TREC systems, fusion resulted in superior precision to that of any of the individual input systems. However, for their own system, it degraded performance. From this, they concluded that fusion of the results of highly-effective retrieval strategies in the same system would not improve effectiveness. They also proposed an alternative hypothesis for situations where fusion would be beneficial. Their hypothesis is that result sets with a high percentage of unique relevant documents are likely to cause better results with CombMNZ.

It is worth noting that the MAP scores of the TREC systems outperformed that of Beitzel's own highly-accurate techniques. This can be taken to suggest that it is possible to introduce systemic differences into a system without adversely affecting performance and that if highly effective techniques that output dissimilar result sets can be used, fusion may still be beneficial.

A variation of CombMNZ was proposed in [63]. Named *WMNZ*, the technique multiplies the sum of the normalised scores by the sum of the weights of the input systems, rather than the number of input systems. The input system weights are calculated in the same way as for WSUM.

Shadow Documents

Wu and Crestani [64] proposed two "Shadow Document" methods of fusion. The focus of their work was partially overlapping databases. In that scenario, algorithms that use a document's presence in multiple result sets as evidence of relevance cannot be used.

Two Shadow Document Methods are proposed. For each technique, the score assigned to each document for fusion purposes is the sum of its normalised scores in each result set. If a document appears in one result set but not another, it is assumed that a "shadow document" of it appears in the one it doesn't itself appear in. This shadow document is assigned a score which is based on its normalised score in the result set it does appear in and also on a coefficient. It is in the calculation of this coefficient that the two techniques vary.

- SDM1 - the coefficient is determined empirically
- SDM2 - the coefficient is a function of the degree of overlap of the result sets.

SDM2 is superior and also outperforms CombMNZ and CombSUM. Both SDM methods are better with a greater degree of overlap, though this effect is strongest for SDM1.

2.4.3.2 Rank-based Techniques

Relevance scores may not be available from all types of input system, particularly in the case of external metasearch. For this reason, some techniques have been developed that take result sets in the form of ranked lists of documents without scores as their input. These are fused using the position of each document in each result set. Some have been compared favourably with popular score-based techniques.

Interleaving and Variations

An early, simple method of merging distinct result sets is *interleaving*, where the results are merged in round-robin fashion [57], whereby the first-ranked documents from each result set are placed at the beginning of the merged set, followed by the second-ranked documents and so on. The effectiveness of this method is largely dependent on the naive assumption that each system returns results of equal quality and an empirical study [56] demonstrates a 40% degradation in effectiveness when compared to the performance of a single centralised collection.

Voorhees et. al. [56] [60] suggested two variations to interleaving. The key focus of both was to use training data to predict which input systems were most likely to return the best results. A greater proportion of higher-ranked systems' result sets were used in the fused result set. Once the documents to be fused had been identified, they were fused in a weighted fashion. For each position in the fused result set, a system was first chosen by rolling a C-sided die, biased by the number of documents remaining in each of the C result sets. Once a system was chosen, the first document remaining in its set was inserted into the fused set. This had the effect of preserving the rankings returned by each individual system and giving priority to those systems judged most likely to return relevant documents.

The two algorithms only differ in their methods of ranking the input systems. Both rely on comparing the given query to training queries. With *Modeling Relevant Document Distributions*, the query is compared (using cosine similarity) with each training query and the number of documents to be taken from each result set is based on the performance of each information source for the queries that are most similar to the given query. *Query clustering* involves creating centroid clusters with the training queries, based on the number of retrieved documents that queries have in common. In this case, the number of documents to take from each result set is based on the performance of each system for the queries in the cluster that the given query is closest to.

Although both of these methods showed vastly superior performance to simple interleaving, Callan et. al. [9] were of the opinion that they are unsuited to widely distributed and dynamic collections and that they are only suited to static collections. Because training information is required for each server, either method would make it more difficult to add new servers and collections to the system at a later date.

Voting

Aslam and Montague have proposed two fusion algorithms that are based on algorithms designed to identify successful candidates in democratic elections where there are more than two candidates. Both fusion techniques are based on algorithms developed in the 18th century to overcome limitations in the simple majority voting system.

Borda-fuse [2] is based on a democratic election model designed for a situation where there are many candidates, but few voters. They use the analogy that the voters are equivalent to the input systems being used and the candidates are represented by the documents retrieved.

With Borda-fuse, each voter ranks a set of c candidates in order of preference. The top ranked candidate is awarded c points, and the score for each candidate decreases by one as one progresses down the list. The total score for any one candidate is the sum of the points awarded to it by all the voters.

A weighted version of this system was also proposed. Here, the scores given by each system to each document are multiplied by a system weight before they are summed. Aslam and Montague used the average precision of each input system as the weight. This was calculated by measuring the performance of training queries run on each system.

The other voting model they proposed is *Condorcet-fuse* [37]. Under the traditional Condorcet algorithm, the winner is the candidate that beats or ties with every other candidate in a pairwise comparison. It must be adapted slightly for fusion, since the goal is not merely to identify the top-ranked document but rather to include all documents in a ranked list.

Condorcet fuse firstly creates a list of all documents returned by any input system. It then uses the QuickSort algorithm with the comparison function being the relative ranking of documents in each system. A document will be ranked above another if it is ranked higher in more of the input result sets.

They also proposed a weighted variation of Condorcet-fuse. Rather than using a simple count of the number of input systems that ranked one document above the other, they use the sum of the weights of those input systems instead. As with weighted Borda-fuse, they use each system's historical mean average precision as a weight. In both cases, they accepted that this may not be the optimal weighting scheme and that its use was to illustrate that using weighting in these algorithms can improve performance

For both algorithms, the weighted variation outperformed the simple algorithm. Weighted Condorcet-fuse was considered to be the best performer. Condorcet-fuse outperformed the standard CombMNZ algorithm in 3 out of 4 test collections, with Weighted Borda-fuse achieving similar performance to CombMNZ.

Because they rely on voting, both of these algorithms are only useful for data fusion where the database is identical, since it must be possible for each document to be returned by each input system.

Rank-based CombMNZ

In [33], as well as using normalised scores, Lee also converted ranks to scores for use with Fox and Shaw's CombMNZ algorithm [18]. The conversion was done using the following function:

$$Rank\_Sim(rank) = 1 - \frac{rank - 1}{num\_of\_retrieved\_docs} \tag{2.16}$$

Although this didn't perform as well as CombMNZ with normalised scores, average precision was within 0.01 in almost all cases. This suggests that in situations where relevance scores are not available, rank-based CombMNZ could be a useful fusion technique.

2.4.3.3 Other Techniques

A number of other techniques have been developed in the area of external metasearch.

The *Inquirus* meta search engine [31] downloads each document returned by the input systems and ranks them based on the number of query terms present in the document, the proximity of those terms and term frequency.

In [14], Craswell et. al. proposed two alternative merging techniques which also involved the result sets being re-ranked based on documents' content. In *Reference Statistics*, instead of ranking documents based on full collection statistics (which may not be available for a meta search engine), they use a system based on a reference-statistic database. This contains relevant statistics on some set of documents which may or may not be a subset of the actual document collection being used (in their experiments, the authors used statistics on 10% of the actual document collection). This means that retrieved documents can be ranked on a full-text basis using a much smaller index. It also caters for situations where underlying engines do not allow access to their own index.

Feature Distance Ranking operates on the desire to reduce the time and cost involved in downloading all the retrieved documents from the underlying search engines in order to perform a ranking algorithm operating on a full-text basis. Instead, a ranking algorithm was developed that gives a higher weighting to a term appearing near the start of the document. This is based on the assumption that terms are less important if they appear at the end of the document, if it doesn't appear in close proximity to other query terms, if it has already appeared earlier in the document or if it is a common term in the collection. Because of this bias in favour of terms appearing earlier in the document, query terms appearing towards the end of the document have a relatively small impact on the ranking score attributed to the document. This means that, rather than having to download the full text of each document, a reliable ranking score can be calculated by using only a portion of each document, thus saving on processing and network costs, consequently reducing the time required to perform the document merging.

A further ranking system was proposed in [20], whereby each server returns a ranked list of documents along with accompanying metadata. The data proposed includes raw-score (the ranking score assigned by the server to each document returned for the query in question),term-frequency (the number of times each query term appears in the document), term-weight (the weight assigned each query term in each document by each server, using whichever weighting algorithm is in use on the server itself), document-frequency (the number of documents on the server that each query term appears in), document-size (the size of the document in bytes) and document-count (the number of tokens, as defined by the server (e.g. the number of terms in the document) contained in the document).

2.5 Chapter Summary

This chapter discusses work that previous researchers have carried out in the area of Information Retrieval and specifically in the area of data fusion. A number of IR models that have been proposed in the past are also discussed. Particular attention has been paid to the Vector Space Model, the Fuzzy Set Model and the Extended Boolean Model, which are used to produce the inputs to the data fusion experiment detailed in Chapter 4.

A number of methods of evaluating the performance of IR systems are also discussed. These will also used for the evaluation of the experiments in Chapters 4, 5 and 6.

Finally, the challenges inherent in the data fusion problem are outlined, along with a description of a number of approaches that have been taken in the past. Of these, the CombMNZ algorithm has become a popular algorithm against which novel data fusion techniques are compared. This will serve as the baseline against which *probFuse* is compared in subsequent chapters.

CHAPTER

THREE

Probabilistic Data Fusion using *probFuse*

3.1 Introduction

This chapter describes *probFuse*, a probabilistic approach to data fusion. *ProbFuse* ranks documents based on the probability that they are relevant to a given query. This probability is calculated during a training phase, and depends on both the input system that returned the document amongst its results and the position in the result set the document was returned in.

The inputs to the fusion process are a number of collections of result sets that are produced by different IR models running the same queries on the same document collection, which means that *probFuse* falls under the category of data fusion. Data fusion is outlined in Section 2.4.1.2, along with an example of a real-world use for it. In order to run *probFuse*, it is first necessary to build a set of probabilities for each input system. These probabilities are calculated by analysing the performance of each individual model on a number of training queries.

For each of these training queries, the result sets are divided into a number of segments. Based on the proportion of documents in each segment that are relevant to the training query, the probability that a document appearing in that segment is relevant to a given query is calculated. This probability is averaged over all the training queries, resulting in each input system being assigned an average probability value for each segment. The division of result sets into segments is explained in Section 3.2. The calculation of the probability of relevance for each segment is explained in Section 3.3.

Once these probabilities have been calculated for each input system, *probFuse* can then fuse subsequent result sets returned by those systems. This process is explained in Section 3.4.

3.2 Segmentation

Many fusion algorithms calculate a relevance score for a document based on either the individual relevance scores assigned to it in each input result set or the rank of that document in each of these inputs (See Section 2.4.3).

Rather than using either of these approaches, *probFuse* divides each result set into a number of "segments". The number of segments each result set is divided into is denoted by x. Values of x that result in best performance are determined empirically (see Chapters 4 and 5).

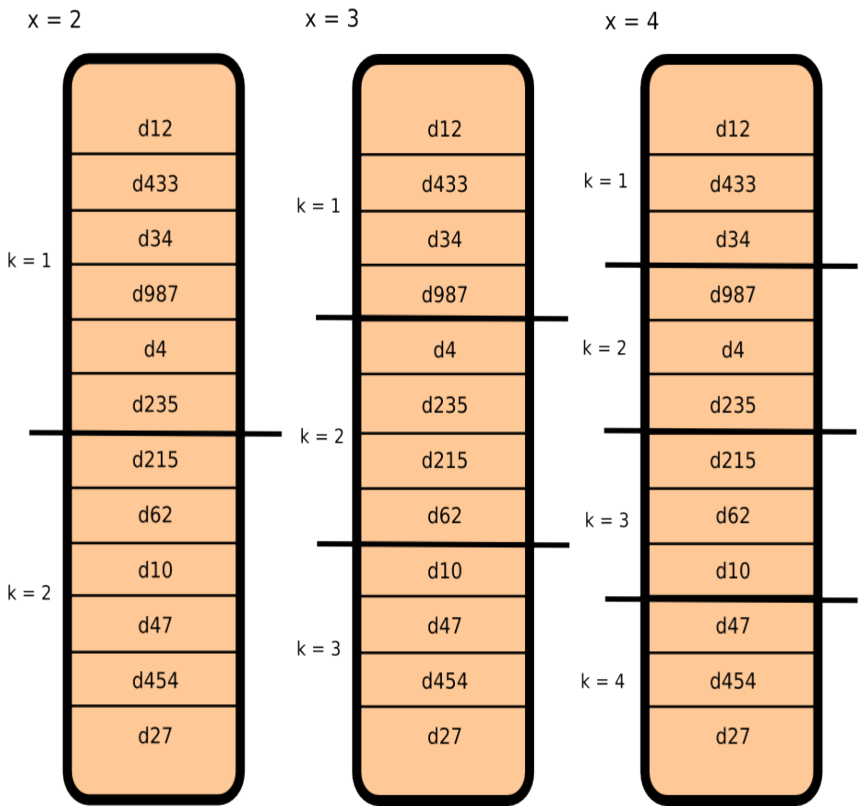

Figure 3.1: Segmenting a result set for different values of x

Figure 3.1 shows an example of segmenting a result set containing twelve documents using three different values of x. Taking document *d215* as an example, it can be seen that for a value of $x = 2$ (i.e. the result set is divided into two segments), it appears in the second segment of the result set. For $x = 3$, it is still in the second segment, moving to the third for $x = 4$.

One motivation for the use of segments rather than just the rank a document appears in is the possibility of input systems returning result sets of different lengths in response to different queries. For example, a document ranked 10th in a 20-document result set is less likely to be relevant than one ranked 10th in a result set of 200 documents.

In addition, *ProbFuse* requires a training phase to calculate probabilities of relevance for each segment. If these probabilities had been calculated based on document rank, performing fusion on a result set that contains more documents than the training result sets would be problematic. This is because no probabilities would be available for the documents located at the end of the result set.

The number of segments each result set is divided into remains constant throughout the fusion process. However, the number of documents contained in each segment changes in proportion to the length of the result set. Thus, each result set is always divided into x segments, and x probability values are also available.

3.3 Probability Calculation

The training phase of *probFuse* involves calculating the probability that a document being returned by a particular input system in a particular segment is relevant to a given query. The calculation of this probability is carried out by using t% of all the available queries for training purposes. Values of t that result in the best retrieval performance are determined empirically (see Chapters 4 and 5).

In order to make use of the results of these training queries, relevance judgments included with the relevant document collection are utilised. For each query, a list of which documents in the collection are relevant or nonrelevant to that query is included. For some, generally small, document collections, complete relevance judgments are available. This means that for each available query, all the documents in the collection have been judged to be either relevant or nonrelevant to that query. However, for larger collections, complete relevance judgments are not always available, so the relevance of many documents is unknown.

Because of this variation in the completeness of relevance judgments, two slightly different methods of calculating the probability of relevance are proposed. *ProbFuseAll* considers unjudged documents to be nonrelevant and is described in Section 3.3.1. *ProbFuseJudged* only takes judged documents into account, ignoring unjudged documents. It is described in Section 3.3.2.

3.3.1 ProbFuseAll

The probability of relevance for documents in each segment is calculated by considering the fraction of all the documents in the segment that are relevant to the given query. This probability calculation is referred to as *probFuseAll*, in order to distinguish it from the *probFuseJudged* variation described below in Section 3.3.2.

In a training set of Q queries, $P(d_k|m)$, the probability that a document d returned in segment k is relevant, given that it has been returned by retrieval model m, is given by:

$$P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{|R_{k,q}|}{|k|}}{Q} \tag{3.1}$$

where $|R_{k,q}|$ is the number of documents in segment k that are judged to be relevant to query q, and $|k|$ is the total number of documents in segment k.

The reason for averaging this probability over a number of training queries is to enable an accurate representation of the typical distribution of relevant documents to be built for each input system. This will be used during the fusion phase to assign a final score for each document that will determine its rank within the fused result set.

3.3.2 ProbFuseJudged

ProbFuseJudged is a variation of *probFuseAll* that only takes judged documents into account when calculating probabilities. Documents that have not been judged are ignored.

The formula itself is quite similar to that of *probFuseAll*, with the only adjustment being that the probability is calculated by finding the fraction of judged documents in each segment that are relevant, rather than the fraction of all documents.

For *probFuseAll*, the probability $P(d_k|m)$ is given by

$$P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{|R_{k,q}|}{|R_{k,q}|+|N_{k,q}|}}{Q} \tag{3.2}$$

where $|R_{k,q}|$ is the number of documents in segment k that are judged to be relevant to query q, and $|N_{k,q}|$ is the number of documents in segment k that are judged to be nonrelevant to query q. Segments containing only unjudged documents are skipped, as both $|R_{k,q}|$ and $|N_{k,q}|$ would be zero, which would cause the probability to be undefined.

From this equation, it can see that for document collections with complete relevance judgments, probabilities calculated using *probFuseJudged* will be identical to those calculated using *probFuseAll*. This is because all documents will have either been judged relevant or judged nonrelevant, meaning that there are no unjudged documents to be ignored.

More formally, this can be expressed by

$$|U_{k,q}| = 0 \Rightarrow |k| = |R_{k,q}| + |N_{k,q}| \tag{3.3}$$

where $|U_{k,q}|$ is the number of documents in segment k that have not been judged either relevant or nonrelevant to query q, $|k|$ is the total number of documents in segment k, and $|R_{k,q}|$ and $|N_{k,q}|$ are the number of documents in segment k that have been judged relevant and nonrelevant to query q respectively.

3.3.3 Example Probability Calculation

Figure 3.2 and Tables 3.1 and 3.2 show an example of calculating the probability of relevance figures for each segment for one input system. Specifically, Figure 3.2 shows three training result sets, (a), (b) and (c), that are segmented based on an x-value of 4. Documents named R_x are documents that have been judged to be relevant to the training query in question. N_x indicates that the document has been judged nonrelevant, while U_x represents documents that are unjudged.

Probabilities calculated using *probFuseAll* are shown in Table 3.1, while those of *probFuseJudged* are contained in Table 3.2.

Taking the first segment as an example ($k = 1$), it can be seen that result set (a) returns three relevant documents in that segment. In this case, *probFuse-*

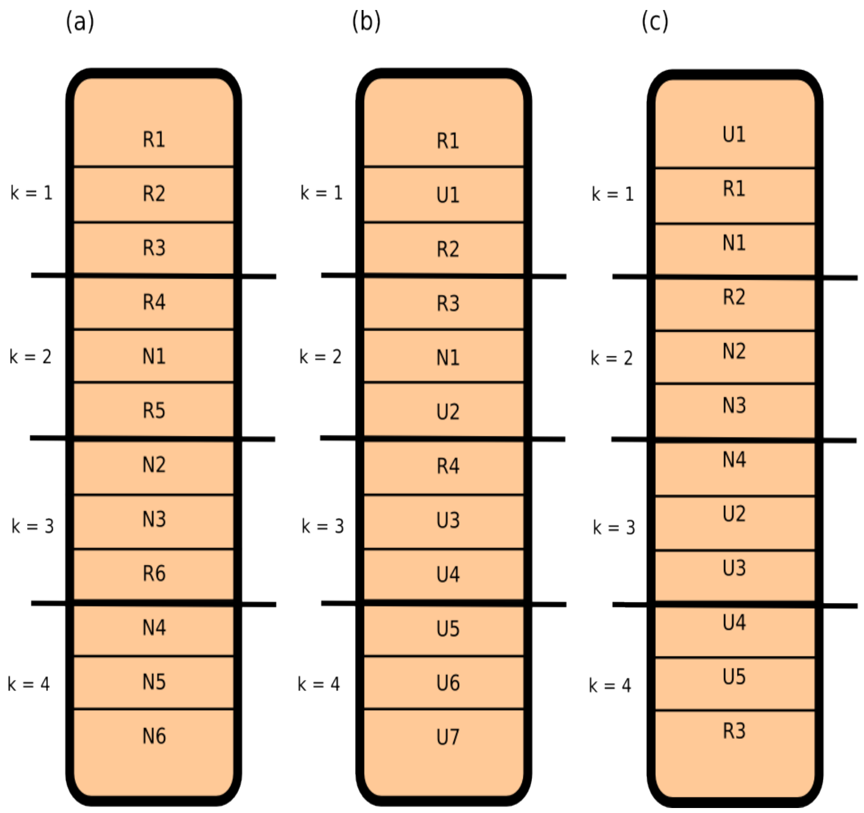

Figure 3.2: Result sets for example probability calculations

All will calculate a probability of 1.0, since all the documents in the segment are judged relevant. The probability value calculated by *probFuseJudged* will also be 1.0, since all the judged documents in the segment are judged relevant. For result set (b), *probFuseAll* returns a probability of 0.66, since there are two relevant documents from three documents in total. *ProbFuseJudged* ignores the unjudged document that is returned in second place in the result set. Again, it calculates the probability to be 1.0, since all the judged documents in the result set are judged relevant. Result set (c) includes one judged relevant document, one judged nonrelevant document and one unjudged document in the first segment. Here, the probability value calculated by *probFuseAll* will be 0.33, as only one of the three documents in the segment is judged relevant. *ProbFuseJudged* ignores the unjudged document and produces a probability of 0.50 since one of the two judged documents is judged to be relevant.

In this example, result set (a) does not contain any unjudged documents. This was done to demonstrate that the probabilities calculated by *probFuseAll* and *probFuseJudged* are the equal when complete relevance judgments are available. Whenever unjudged documents are present in a segment, the probability calculated using *probFuseJudged* will be higher than that of *probFuseAll*.

Table 3.1: Probability calculation example using *probFuseAll*

k	(a)	(b)	(c)	Average
1	1.00	0.67	0.33	0.67
2	0.67	0.33	0.33	0.44
3	0.33	0.33	0.00	0.22
4	0.00	0.00	0.33	0.11

Table 3.2: Probability calculation example using *probFuseJudged*

k	(a)	(b)	(c)	Average
1	1.00	1.00	0.50	0.83
2	0.67	0.50	0.33	0.50
3	0.33	1.00	0.00	0.44
4	0.00	N/A	1.00	0.50

Segment 4 of result set (b) is an example of a segment that is ignored by *probFuseJudged* as a result of it only containing unjudged documents. In this case, the final probability of relevance for segment 4 is the average of result sets (a) and (c), as (b) does not provide any information on judged documents in that segment. In contrast, *probFuseAll* assumes all unjudged documents to be non-relevant, so it assigns a probability of zero to that segment, which is taken into account for the average.

In order to be able to proceed to the fusion phase, a set of probabilities such as those shown in Tables 3.1 and 3.2 needs to be created for each input system.

3.4 Fusion

Once the training phase has been completed and a set of probabilities of relevance has been created for each of the input systems, fusion may be performed on the results of subsequent queries.

For these queries, the ranking score S_d for each document d is given by

$$S_d = \sum_{m=1}^{M} \frac{P(d_k|m)}{k} \tag{3.4}$$

where M is the number of retrieval models being used, $P(d_k|m)$ is the probability of relevance for a document d_k that has been returned in segment k in retrieval model m, and k is the segment that d appears in (1 for the first segment, 2 for the second, etc.). For any input system that does not return document d in its result set at all, $P(d_k|m)$ is considered to be zero, in order to ensure that documents do not receive any boost to their ranking scores from models that do not consider them to be relevant.

This approach to data fusion attempts to make use of the three effects described in Section 2.4.3. By using the sum of the probabilities, more significance is attached to documents that have been returned by multiple input systems, thus exploiting the Chorus Effect. The division by the segment number k gives a greater weight to documents that appear early in each of the individual result sets, making use of the Skimming Effect. Finally, because the probabilities are calculated based on the actual past performance of each input system, greater importance is attached to input systems that are more likely to return relevant documents in particular segments (Dark Horse Effect).

3.4.1 Fusion Example

Figure 3.3 shows an example of three input result sets which are to be fused, along with the final fused result set. Each of these inputs is from a different input system, named "one", "two" and "three". Sample probability values for each are provided in Table 3.3.

Using the given input result sets and the sample probabilities provided, Table 3.4 illustrates how the fusion process works. The leftmost column includes the document identifiers, ordered by their position in the final result set. The columns marked "one", "two" and "three" show the probability calculation of each document for each of the input systems. Finally, the column titled "total" contains the final score that is used to rank the fused result set.

For example, document $d1$ occurs in the third segment of result set "one" and the first segment in each of "two" and "three". For each result set, the probabil-

Table 3.3: Sample probability data for three input systems

k	one	two	three
1	0.75	0.67	0.90
2	0.67	0.50	0.55
3	0.33	0.30	0.26
4	0.10	0.00	0.15

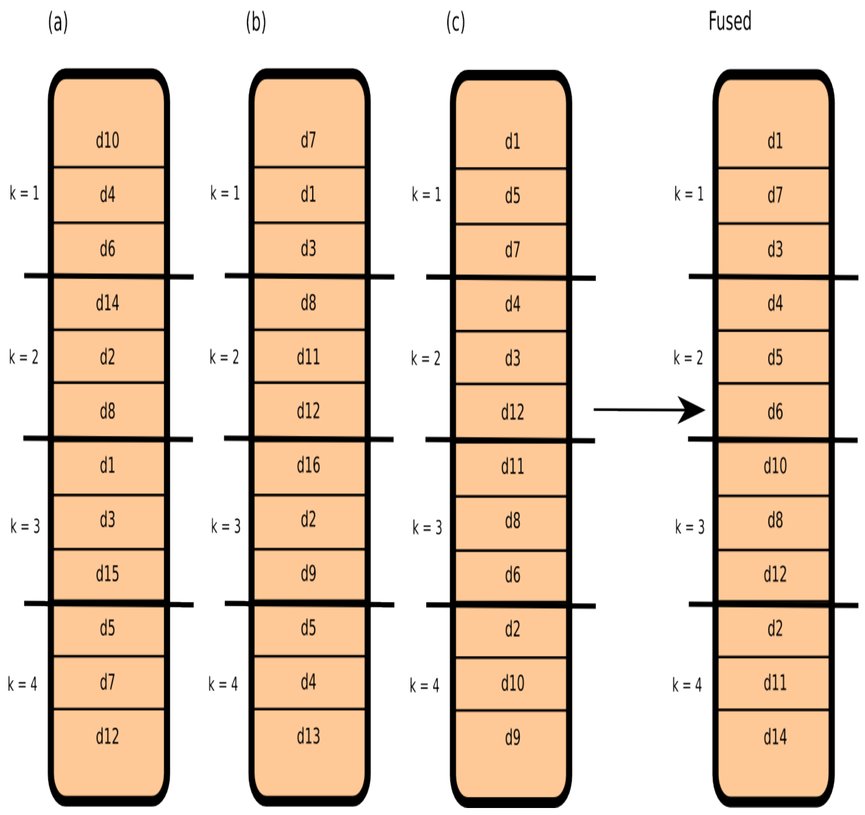

Figure 3.3: Sample input result sets with fused result set

ity score associated with the segment the document was returned in is divided by the segment number. For that reason, the probability for document $d1$ in result set "one" is divided by 3, as it occurred in the third segment, and divided by 1 for the other two result sets. These scores are then added together to produce the final score for ranking in the fused result set.

In Table 3.4, some of the probability calculations are marked as 0 (e.g. for document $d6$ in result set "two"). This indicates that the document in question was not returned in that result set. As indicated in Section 3.4, the probability is assumed to be zero in this situation.

After all the calculations in Table 3.4 have been carried out, the fused result set illustrated in Figure 3.3 is produced.

3.5 Summary

In this chapter, the *probFuse* algorithm is introduced, which is a data fusion algorithm that produces its fused result set by considering the probability that a

Table 3.4: Sample probability data for three input systems

Document	one	two	three	Total
d1	0.330 / 3	0.670 / 1	0.900 / 1	1.680
d7	0.100 / 4	0.670 / 1	0.900 / 1	1.595
d3	0.330 / 3	0.670 / 1	0.550 / 2	1.055
d4	0.750 / 1	0.000 / 4	0.550 / 2	1.025
d5	0.100 / 4	0.000 / 4	0.900 / 1	0.925
d6	0.750 / 1	0	0.260 / 3	0.837
d10	0.750 / 1	0	0.150 / 4	0.787
d8	0.670 / 2	0.500 / 2	0.260 / 3	0.672
d12	0.100 / 4	0.500 / 2	0.550 / 2	0.550
d2	0.670 / 2	0.300 / 3	0.150 / 4	0.472
d11	0	0.500 / 2	0.260 / 3	0.337
d14	0.670 / 2	0	0	0.335
d9	0	0.300 / 3	0.150 / 4	0.137
d15	0.330 / 3	0	0	0.110
d16	0	0.300 / 3	0	0.100
d13	0	0.000 / 4	0	0.000

document is relevant based on its position in the input result sets. It does this by dividing each result set into a number of segments and calculating the probability of relevance in each segment by analysing the result sets produced in response to a number of training queries. The *probFuse* algorithm is designed to make use of the three "effects" outlined in Section 2.4.3.

Two methods of calculating the probability of relevance have been proposed. *ProbFuseAll* considers unjudged documents to be nonrelevant to a given query whereas *probFuseJudged* ignores unjudged documents and only takes judged documents into account.

This chapter also provides examples to illustrate how these probabilities are initially calculated, including two variations: *probFuseAll* and *probFuseJudged*. An example of how this probability data is used to perform fusion on a number of input result sets is also provided.

Optimal values for t (the percentage of available queries uses for training) and x (the number of segments into which to divide each result set) must be determined empirically. To calculate these, and to compare the results with a baseline data fusion technique, three sets of experiments were performed. Chapter 4 describes an exploratory study that investigates the merits of *probFuse* when applied to small document collections with complete relevance judgments. Chapter 5 describes experiments on larger collections, using inputs from the TREC conferences. These aim to show that it is possible to find a training set size and number of segments that performs well on both sets of inputs. Finally, Chapter 6, uses the same training set size and number of segments as Chapter 5 to investigate the effects of running *probFuse* on a collection for which these values have not been specifically optimised.

CHAPTER

FOUR

Exploratory Study on Small Document Collections

4.1 Introduction

This chapter describes an initial exploratory study that was performed to explore the merit of the *probFuse* algorithm on small collections. A key factor in the decision to use small collections initially is the presence of complete relevance judgments. This means that for every document in the collection, it is known whether that document is relevant or nonrelevant to each of the accompanying queries. Because of the need for relevance judgments in the training phase of the *probFuse* algorithm, the presence of complete relevance judgments represents an ideal operating environment in which to initially test the proposed algorithm.

Section 4.2 outlines the aims of this study. The setup of the experiment are described in Section 4.3. Finally, Section 4.4 analyses the results produced by *probFuse*, comparing it with the individual inputs on which it performed fusion and with the baseline CombMNZ technique.

4.2 Experimental Aims

The principal aim of this initial exploratory study is to demonstrate that data fusion using *probFuse* results in superior retrieval performance to any of the individual inputs being used. In order to do this, it must be possible to find values for t (the training set size) and x (the number of segments into which each result set is divided) that result in increased retrieval performance for all

Table 4.1: Characteristics of Document Collections Used

Collection	Documents	Queries
Cranfield	1,400	225
LISA	5,872	35
Med	1,033	30
NPL	11,429	93

document collections used. Therefore, in this experiment, a number of combinations of values for t and x are evaluated in order to identify values that result in the greatest performance increase.

An additional objective is to demonstrate that *probFuse* achieves superior performance to that of the CombMNZ data fusion algorithm.

4.3 Experimental Setup

For this study, four small document collections were chosen for fusion: Cranfield, LISA, NPL and Med. The characteristics of each of these document collections are outlined in Table 4.1. Each of these collections have complete relevance judgments available to them, meaning that every document has been judged to be either relevant or nonrelevant to each query. As discussed in Section 3.3, no distinction is made between *probFuseAll* and *probFuseJudged* in this chapter. This is because the values produced by the probability calculations are equal.

For each of these collections, three different input systems were simulated by running the available queries with three different IR techniques: the Vector Space Model [47], the Fuzzy Set Model [3] and the Extended Boolean Model. These models are outlined in Sections 2.2.2, 2.2.4 and 2.2.5 respectively.

As a baseline against which to compare the performance of *probFuse*, the common CombMNZ algorithm was chosen, which is described in Section 2.4.3.1.

In order to reduce the possibility of the results being affected by the order of the queries, the queries relating to each document collection were first arranged in a random order. Once this was done, each query relating to each of the four document collections was run by each of the three IR techniques. This produced three result sets for each query, each produced by a different IR model. These result sets were used as the inputs for data fusion.

For each query, the result sets were fused by *probFuse* using a number of different combinations of values for t and x. Training set sizes t such that $t \in \{10,20,30,40,50,60,70,80,90\}$ and numbers of segments x such that $x \in \{2,4,6,8,10,20,30,40,50\}$ were used.

The evaluation measure used in this study is based on the interpolated precision/recall curve that was described in Section 2.3.2. This evaluation measure also allows a result set produced by a data fusion algorithm to be compared with the best-performing input result set at every recall level. This is in contrast to making comparisons with the best-performing individual input overall. For example, one input may achieve high precision at low levels of recall, but may not perform well at higher levels of recall. Another input may not achieve high precision at low recall, but may perform better as recall increases. The evaluation measure used allows for the evaluation of the fused result sets by comparing them with the former input at low levels of recall and the latter as recall increases.

Firstly, it is necessary to calculate the interpolated precision for each of the individual input result sets. This results in 11 precision values, one for each of the 11 standard recall levels (0% to 100% inclusive, at intervals of 10 percentage points). The same data is necessary for the fused result set that is being evaluated. Once this is done, $\overline{\Delta P_c}$, the mean difference in precision for document collection c is given by

$$\overline{\Delta P_c} = \frac{\sum_{r=1}^{R} P_{f,r} - MAX(P_{c,r})}{R} \tag{4.1}$$

where R is the number of standard recall levels, $P_{f,r}$ is the precision of the fused result set at recall level r and $MAX(P_{c,r})$ is the maximum precision obtained by any single retrieval model on collection c at recall level r. The single value used in Figures 4.1 and 4.2 is the average $\overline{\Delta P_c}$ across all four collections.

4.4 Analysis

Figure 4.1 shows the mean change in precision for the various values of t and x. Each line represents the mean change in precision over each of the four document collections for a different training set size. This will enable values for t and x that perform well on all four collections to be identified.

From this figure, it is observed that the poorest performance is seen at training set sizes of 90% and 10%. This suggests that the use of training sets that are very large or very small will result in poor performance. At a training set size of 50%, the highest mean difference precision is seen for most values of x. The only values for which it fails to achieve the best performance are for $x = 2$, where greater performance is observed for a training set size of 20%, and $x = 10$, where it is outperformed by the training set of 70%.

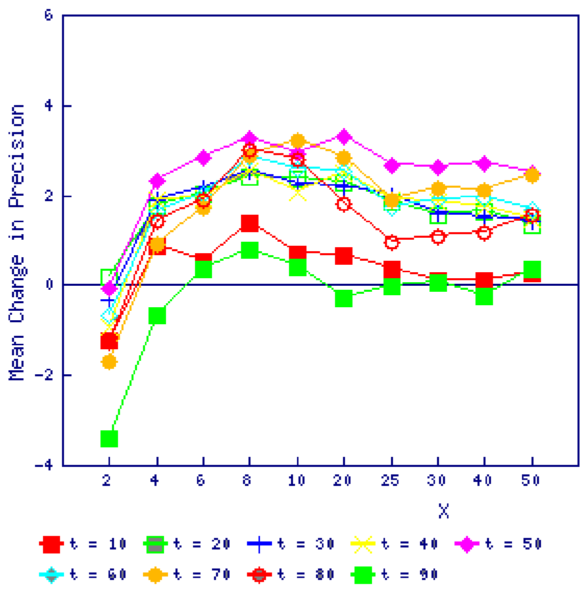

Figure 4.1: Mean difference in precision for different training set sizes

In Figure 4.2, each line represents the change in average precision for a particular value of x. This figure shows that the poorest performance is observed for a value of $x = 2$. This is a predictable outcome, since by dividing the result set into two segments, probability of relevance will be assigned to a document based on which half of the result set it appears in. This is clearly too coarse a measure, as the results show. Initially, increasing values for x produce superior results, with x values of 10 and 20 showing the highest mean precision increases. However, once x increases further, the mean difference in precision begins to decline.

From Figures 4.1 and 4.2, it is observed that the best average performance over each of the four document collections is achieved by using a training set size of 50% and by dividing each segment into 20 segments.

Having identified the combination of x and t values that performs best overall, the performance of *probFuse* on each of the document collections can be evaluated separately. The results achieved by CombMNZ are also presented.

Table 4.2 shows the mean difference in precision for both *probFuse* and

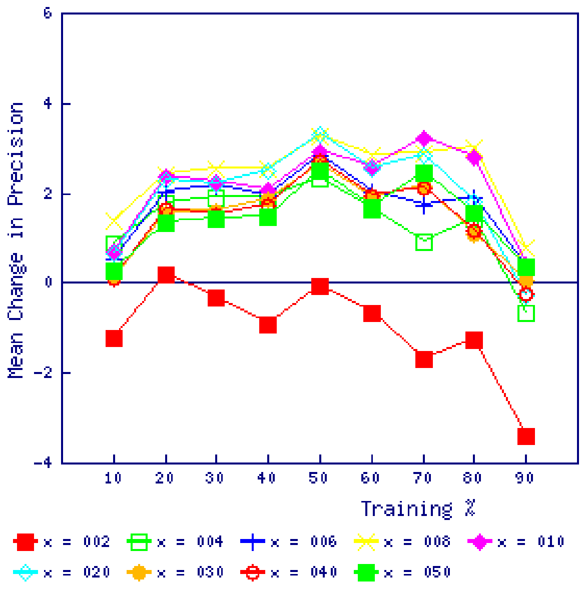

Figure 4.2: Mean difference in precision for different values of x

Table 4.2: Comparison of the mean difference in precision achieved by the *probFuse* and CombMNZ algorithms for each collection

	probFuse	CombMNZ
Cranfield	+1.92**	-1.48*
LISA	+3.09**	+2.24
Med	+3.48	+3.07
NPL	+4.80**	+4.13**
Max	+4.80	+4.13
Min	+1.92	-1.48
Avg	+3.32	+1.99

CombMNZ on each of the document collections. The values for *probFuse* are those produced by using a training set size of 50% and an x-value of 20. In order to enable a valid comparison to be made, CombMNZ was only used on the queries that were used by *probFuse* in its fusion phase. Those queries used by *probFuse* for training purposes were ignored for the purposes of CombMNZ. Entries marked with "*" are statistically significant for a significance level of 5%. Entries marked with "**" are statistically significant for a significance level of 1%, as calculated by the Wilcoxon test [61].

On each of the four document collections, *probFuse* has achieved a higher increase in precision than CombMNZ. *ProbFuse* shows mean increase in precision for all four collections, with these increases being statistically significant for the Cranfield, LISA and NPL collections at a significance level of 1%.

In contrast, CombMNZ achieves a significant increase on the NPL collection alone, and shows a decrease in precision for the Cranfield collection.

Figure 4.3 illustrates the performance of *probFuse* and CombMNZ on the Cranfield collection. It shows the interpolated precision at the standard recall levels for each of the three individual techniques, as well as for each of the two fusion techniques. This figure shows that *probFuse* achieves higher interpolated precision at each of the standard recall levels. This is particularly true for low levels of recall. The achievement of high precision at low levels of recall is a promising result, as this is where users will tend to expect relevant documents to occur.

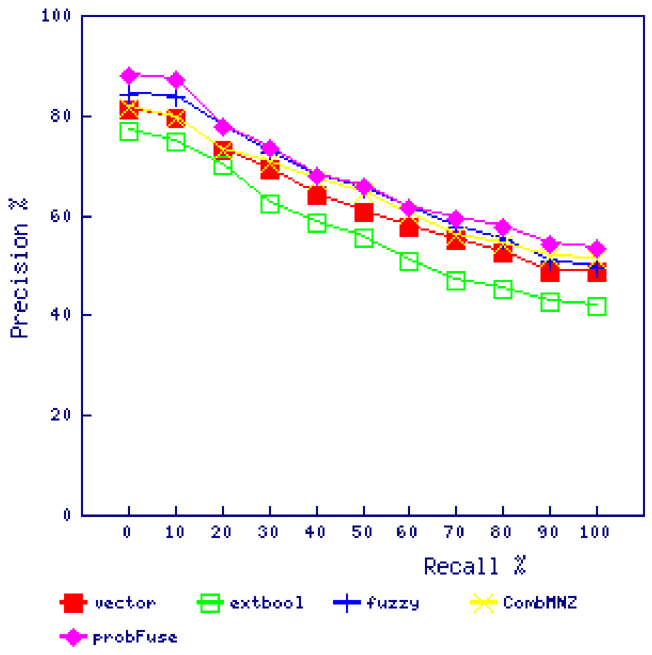

Figure 4.3: Interpolated Precision graph for the Cranfield collection

4.5 Summary

This chapter describes an initial exploratory study to investigate the merit of the *probFuse* algorithm. This study consisted of performing data fusion on a four small document collections for which complete relevance judgments are available. It was found that by using 50% of the available queries for training purposes and by dividing each result set into 25 segments, *probFuse* achieved statistically significant performance improvements over the individual inputs that it was fusing for three of the four document collections, and an average improvement on the fourth. CombMNZ, a data fusion algorithm commonly used as a baseline, was also shown to perform better than the individual inputs in the majority of cases, although it was shown to cause a significant decrease in performance for the Cranfield document collection. For each of the collections for which CombMNZ achieved a performance increase, this increase was less than that of *probFuse*.

Having observed promising results on these small document collections, experiments on much larger document collections for which relevance judgments are incomplete may now be performed. These experiments begin in Chapter 5, using data from the TREC-3 and TREC-5 conferences and continues in Chapter 6 with experiments on the web track of the TREC-2004 conference.

CHAPTER

FIVE

Applying *probFuse* to TREC-3 and TREC-5

5.1 Introduction

In Chapter 4, it was demonstrated that *probFuse* shows promise when applied to small document collections for which complete relevance judgments are available. This chapter, performs a more detailed analysis of its performance on larger datasets. These experiments involve performing data fusion on data taken from the ad hoc track of the TREC-3 [24] and TREC-5 [58] conferences.

The Text REtrieval Conference (TREC) began in 1992 and is co-sponsored by the National Institute of Standards and Technology (NIST) and the U.S. Department of Defence. Participants are invited to take part in a number of "tracks", each of which relate to a different type of IR task. The experiments presented in this chapter use data from the ad hoc retrieval task, in which participants are furnished with a document collection and a number of queries (known as "topics"). Each participant group then uses its own IR system to produce a result set for each topic. The quality of these result sets is evaluated by NIST, after which a workshop is held with the aim of allowing participants to share their experiences.

For the TREC-3 and TREC-5 ad hoc datasets, the document collection consisted of mostly news and magazine articles. Relevance judgments are incomplete, meaning that the relevance of all documents is not known for each query. This lack of complete relevance judgments allows *probFuseAll* and *probFuseJudged* to be compared with one another, in addition to comparing each with the standard CombMNZ approach.

The TREC-3 dataset was chosen because it is the dataset on which Lee demonstrated the effectiveness of CombMNZ [33]. The addition of the TREC-5 dataset was motivated by the desire to demonstrate that single values for the training set size and the number of segments can be shown to result in effective data fusion on multiple input sets. The TREC-5 data represents a document collection that is similar in size to that of TREC-3 and which has been used by other fusion researchers in the past [55] [63].

Section 5.2 describes the motivation for the experiments presented in this chapter. In Section 5.3, the setup of these experiments is described, along with details of the inputs that were used. Sections 5.4 and 5.5 show how the optimal training set size and the number of segments to divide each result set into were empirically identified. Finally, Section 5.6 shows the evaluation and analysis of these fusion experiments on each of the input datasets.

5.2 Experimental Aims

The aim of the experiments presented in this chapter is to find optimal values for t (the percentage of available queries to be used for training purposes) and x (the number of segments into which to divide each segment). Ideally, these should enable *probFuse* to achieve superior performance to that of the CombMNZ technique.

The principal objective of this chapter is to identify single values for t and x that will achieve increased performance. In order for *probFuse* to be effective, it must be capable of achieving high performance without it being necessary to alter the number of segments for each document collection. For this reason, the goal of identifying t and x is to find values that will cause *probFuse* to outperform CombMNZ on each of the collections.

5.3 Experimental Setup

The inputs to the fusion experiments presented in this chapter consisted of result sets submitted to the ad hoc retrieval track of the TREC-3 and TREC-5 conferences. Details of the data available in each of these datasets is presented in Table 5.1. Each of these inputs to the fusion process represents the output of one Information Retrieval system when running specific queries on a given document collection. Each input takes the form of a topfile, which is a collection of result sets, each of which is returned in response to a different query (or topic). Each topfile contains a result set for each of 50 queries. The input systems for TREC-3 used TREC topics 151-200, while topics 251-300 were used for TREC-5.

Table 5.1: Details of the TREC-3 and TREC-5 ad hoc datasets

	TREC-3	TREC-5
Number of Documents	741,856	524,929
Collection Size	2Gb	2Gb
Number of Participants	40	31
Number of Topics (Queries)	50	50
Average Number of Documents Judged per Topic	1,946.38	2,673.62

Each group that submitted results to TREC-3 or TREC-5 were given 50 queries on which to test their IR system. Each participating IR system produced a topfile containing a result set for each of these queries. 40 topfiles are available for TREC-3, while 31 are available for TREC-5. For the TREC-5 ad hoc task, there were two categories of participants: Category A participants used of all the available data, whereas Category B was aimed at exploratory groups that wished to evaluate novel retrieval techniques but did not have the experience or resources to deal with large datasets. These groups operated on only a subset of the document collection. For the experiments discussed in this chapter, only Category A participants were considered.

Five experimental runs were performed for each of the two datasets. For each run, six input topfiles were selected at random. No input was selected for multiple runs. Because of the necessity of a training phase, it is possible that the order of the queries may have an effect on fusion performance. In order to counteract this, five random orderings of the input queries were created and fusion was performed on each. The evaluation values presented in Section 5.6 for each run are the average scores over each of these five orderings. The inputs selected for the TREC-3 experiments are shown in Table A.1 and the inputs from TREC-5 are shown in Table A.2. These tables show the filenames of the topfiles that were used as inputs to the fusion process.

Having organised the inputs, fusion was then performed on each set of inputs, using both the *probFuseAll* and *probFuseJudged* variations of the *probFuse* algorithm. The same inputs were then fused using the CombMNZ technique, in order to enable comparisons to be made later.

The initial aim of these experiments was to empirically identify the optimal training set size and the optimal number of segments to divide each result set into. As a result *probFuse* was run for a variety of training set sizes and number of segments. Specifically, training set sizes, t such that $t \in \{10, 20, 30, 40, 50\}$ and numbers of segments, x such that $x \in \{2, 4, 6, 8, 10, 15, 20, 25, 30, 40, 50, 100, 150, 200, 250, 300, 400, 500\}$ were used.

Any result sets used in the training phase for *probFuse* were ignored when fusing with CombMNZ. This is to ensure that the results of the fusion using both algorithms will be comparable, as they will have performed fusion on the same inputs. CombMNZ does not require any training phase, as it operates purely on the similarity scores assigned to each document by each input system. Thus the training queries can safely be skipped.

Because the input data is taken from the TREC conferences, it was decided to evaluate the fused output result sets using *trec_eval*, which is the evaluation software used for the TREC conferences [59]. Two evaluation measures are used in these experiments. Initially, the MAP score is used to find the most effective training set size. Having identified this, the effects of varying the x-value is examined, in order to identify a combination of t and x that performs well for both the TREC-3 and TREC-5 inputs. MAP is described in Section 2.3.3.

Once optimal values for t and x have been identified, the bpref measure is introduced to supplement MAP when analysing the performance of the fusion algorithms. This allows a more detailed comparison between *probFuse* and CombMNZ. The bpref measure is described in Section 2.3.4.

5.4 Training Set Size

This section identifies a training set size that achieves high performance on both the TREC-3 and TREC-5 inputs. Table 5.2 shows the average MAP scores achieved in the experiments on the TREC-3 dataset. Table 5.3 shows the same data for the TREC-5 experiments. Values in bold type are the maximum in their respective columns. Each table also shows the difference between the average MAP score achieved by *probFuse* and that produced by CombMNZ, expressed as a percentage of CombMNZ's score. "CV" indicates the Coefficient of Variation for each column [17]. The Coefficient of Variation is defined as the Standard Deviation divided by the Mean. It measures the degree by

Table 5.2: TREC-3 Average MAP scores for various training set sizes

Training t%	CombMNZ	*probFuse All*	Difference v. MNZ	*probFuse Judged*	Difference v. MNZ
10%	0.29593	0.33885	+14.50%	0.33988	+14.85%
20%	0.29738	0.34146	+14.82%	0.34312	+15.38%
30%	**0.30134**	**0.34628**	+14.91%	**0.34830**	+15.58%
40%	0.29753	0.34307	+15.31%	0.34517	+16.01%
50%	0.29557	0.34230	**+15.81**%	0.34445	**+16.54**%
CV	0.00921	0.00933	0.03991	0.01068	0.04859

Table 5.3: TREC-5 Average MAP scores for various training set sizes

Training t%	CombMNZ	*probFuse All*	Difference v. MNZ	*probFuse Judged*	Difference v. MNZ
10%	**0.17842**	0.26011	+45.79%	0.25987	+45.65%
20%	0.17604	0.25959	**+47.46**%	0.26020	+47.81%
30%	0.17528	0.25842	+47.43%	0.25937	**+47.98%**
40%	0.17720	0.25959	+46.50%	0.26056	+47.04%
50%	0.17712	**0.26061**	+47.14%	**0.26175**	+47.78%
CV	0.00814	0.00376	0.01822	0.00413	0.02443

which the MAP scores vary relative to the mean. The MAP scores in these tables are the average of the MAP scores for every fused result set produced using the relevant training set size. This includes the result sets for each value of x for each fusion run.

At this stage of the analysis, the average MAP score is intended to be a general indicator of overall performance for each training set size. Because it includes the MAP scores for all values of x, it is of limited use for detailed performance analysis. Summarising the performance of every result set produced at each training set size enables the selection of a training set size to use for further analysis. This value is then used when analysing the effect of changing the value of x.

Table 5.2 indicates that for the TREC-3 inputs, the greatest improvement over CombMNZ is achieved at a training set size of 50% for both *probFuseAll* and *probFuseJudged*. Each variant achieves its highest overall average MAP score at a training set size of 30%. On the TREC-5 dataset, *probFuseAll* shows its greatest improvement over CombMNZ at a training set size of 20%, whereas *probFuseJudged* achieves its greatest improvement at 30%. In both cases, however, the effect on performance of varying the size of the training set is quite small.

It is interesting to note that both *probFuseAll* and *probFuseJudged* show an overall improvement over CombMNZ at every training set size for both input datasets. At this stage, no effort has been made to select only those runs that achieve the best performance. The MAP scores shown in these tables are independent of the values for x that were used, since they are averaged over all the runs that were performed using each fusion technique. This means that this average improvement over CombMNZ is despite the inclusion of the poorest performing experimental runs in the average MAP values.

For the purposes of Section 5.5, a single training set size must be chosen in order to be able to identify a high-performance x-value. At a training set size of 50%, both *probFuseAll* and *probFuseJudged* achieve either their highest overall average MAP score or their greatest improvement over CombMNZ for both the TREC-3 and TREC-5 inputs. For the TREC-3 inputs, both achieve their highest overall average MAP score, whereas for TREC-5, the greatest increases over CombMNZ are achieved. On the TREC-5 data, the average MAP score achieved at the training set size of 50% is within 2% of the maximum average MAP achieved for both *probFuseAll* and *probFuseJudged*.

From each of the tables, it can be seen that changing the training set size does not have a large effect on the average MAP scores. This is shown by the Coefficient of Variation being less 0.02 for the average MAP score and less than 0.05 for the improvement over CombMNZ on the TREC-3 dataset. This is lower for the TREC-5 inputs, for which the Coefficient of Variation is less than 0.005 for the average MAP score and less than 0.03 for the improvement over CombMNZ.

5.5 Number of Segments

Having determined that best overall performance is achieved for a training set size of 50%, it is necessary to identify an optimal value for x, the number of segments into which each result set is divided. This is done by examining the MAP scores achieved by using each of the x-values listed in Section 5.3 with a training set size of 50%.

Figures 5.1 and 5.2 show the MAP scores achieved by using different values for x, the number of segments each result set is divided into. These represent the results from the TREC-3 and TREC-5 input sets respectively. Each of these MAP scores is the average for each of the five fusion runs. Individual graphs for each of the five runs can be seen in Appendix B for TREC-3 and appendix C for TREC-5.

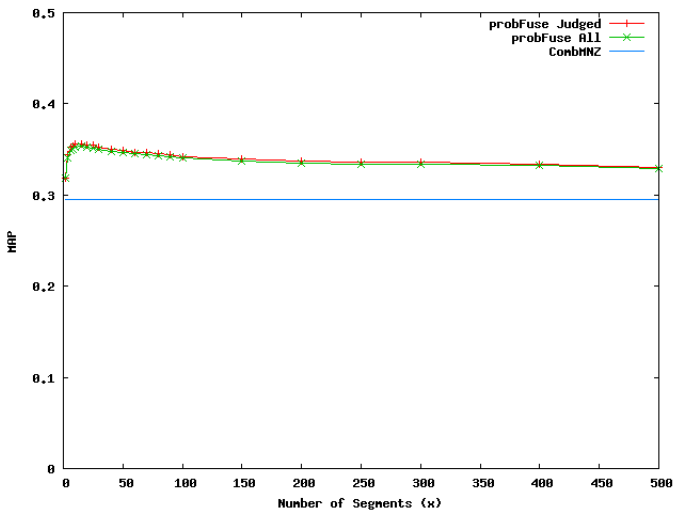

Figure 5.1: TREC-3 MAP scores for $t = 50\%$ (average over 5 runs)

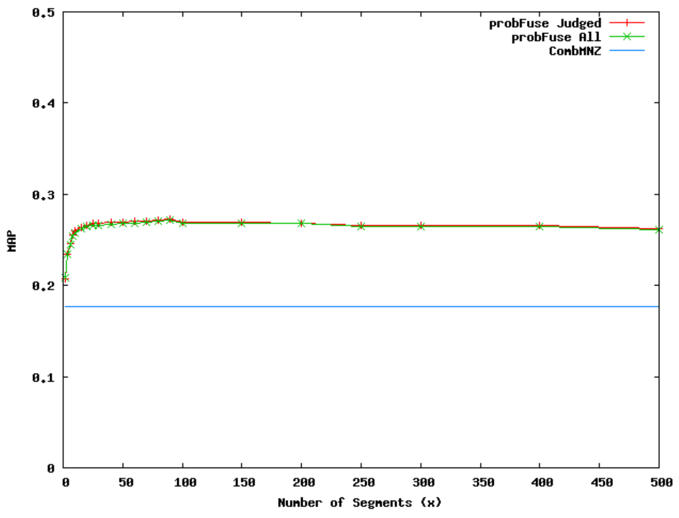

Figure 5.2: TREC-5 MAP scores for $t = 50\%$ (average over 5 runs)

For both input sets, performance is at its worst for a value of $x = 2$. For this value, result sets are divided in two, so the probability of relevance is based on whether a document appears in the first or second half of the result set. It is unsurprising that this value would produce the lowest MAP scores, as it is quite a coarse measure. Both input sets show performance increasing as x increases initially, while both show MAP decreasing slightly at higher values of x. *ProbFuseAll* and *probFuseJudged* show almost identical results, with

Table 5.4: TREC-3 performance of five individual runs for $t = 50\%$ and $x = 25$ using *probFuseAll* and *probFuseJudged*

	CombMNZ		*probFuseAll*	
	MAP	bpref	MAP	bpref
first	0.16726	0.23960	0.30988 (+85.27%)	0.31458 (+31.29%)
second	0.28752	0.33434	0.34100 (+18.60%)	0.33118 (-0.95%)
third	0.43344	0.41222	0.41348 (-4.61%)	0.39222 (-4.85%)
fourth	0.23416	0.31048	0.30374 (+29.71%)	0.30314 (-2.36%)
fifth	0.35548	0.39616	0.39108 (+10.01%)	0.38006 (-4.06%)
Average	0.29557	0.31707	0.35184 (+19.04%)	0.34804 (+9.77%)

	CombMNZ		*probFuseJudged*	
	MAP	bpref	MAP	bpref
first	0.16726	0.23960	0.31144 (+86.20%)	0.31628 (+32.00%)
second	0.28752	0.33434	0.34402 (+19.65%)	0.33356 (-0.23%)
third	0.43344	0.41222	0.41620 (-3.98%)	0.39416 (-4.38%)
fourth	0.23416	0.31048	0.30766 (+31.39%)	0.30528 (-1.67%)
fifth	0.35548	0.39616	0.39294 (+10.54%)	0.38308 (-3.30%)
Average	0.29557	0.31707	0.35445 (+19.92%)	0.35046 (+10.53%)

probFuseJudged being slightly higher at most points.

The point at which MAP begins to decline is different for the two input datasets. For TREC-3, the MAP is at its peak for a value of $x = 15$, after which it declines as x increases. The scores for the TREC-5 inputs continue to increase until $x = 90$, at which point the downward trend seen for higher x-values on TREC-3 is observed.

Dividing each result set into 25 segments yields the best average MAP score for both TREC-3 and TREC-5. Given that the goal of this section is to identify a single value for x that will yield high performance for both input datasets, this is the value that was chosen for analysing the individual runs in Section 5.6.

5.6 Analysis of Individual Runs

Table 5.4 shows the evaluation of the fused result sets produced by running *probFuseAll* and *probFuseJudged* on the TREC-3 inputs. The values in parenthesis are the percentage difference from the corresponding score achieved by the CombMNZ algorithm. Each table shows the MAP and bpref scores for each of five runs, which are named "one", "two", "three", "four" and "five".

Table 5.5: TREC-5 performance of five individual runs for $t = 50\%$ and $x = 25$ using *probFuseAll*

	CombMNZ		*probFuseAll*	
	MAP	bpref	MAP	bpref
first	0.25144	0.26406	0.27378 (+8.88%)	0.26814 (+1.55%)
second	0.22480	0.26896	0.35560 (+58.19%)	0.33918 (+26.11%)
third	0.12306	0.19232	0.26838 (+118.09%)	0.24744 (+28.66%)
fourth	0.12626	0.14520	0.15546 (+23.13%)	0.15734 (+8.36%)
fifth	0.16004	0.21790	0.27842 (+73.97%)	0.26474 (+21.50%)
Average	0.17712	0.19740	0.26633 (+50.37%)	0.25537 (+29.37%)

	CombMNZ		*probFuseJudged*	
	MAP	bpref	MAP	bpref
first	0.25144	0.26406	0.26264 (+4.45%)	0.26878 (+1.79%)
second	0.22480	0.26896	0.35844 (+59.45%)	0.34140 (+26.93%)
third	0.12306	0.19232	0.27050 (+119.81%)	0.24920 (+29.58%)
fourth	0.12626	0.14520	0.15602 (+23.57%)	0.15746 (+8.44%)
fifth	0.16004	0.21790	0.27922 (+74.47%)	0.26498 (+21.61%)
Average	0.17712	0.19740	0.26787 (+51.24%)	0.26212 (+32.79%)

Using the MAP evaluation measure, it can be seen that both *probFuseAll* and *probFuseJudged* outperform CombMNZ for four of the five runs. The only exception is “third”, where CombMNZ achieves a slightly higher MAP score. It is important to highlight that the MAP scores produced by the *probFuse* algorithms are actually higher than those achieved for any of the other four runs. CombMNZ also achieves its highest MAP score on this run. This, combined with the fact that *probFuse* outperforms CombMNZ on every other run, suggests that the “third” run is an outlier that can be considered to be an exception to the trend. The average MAP score of the two *probFuse* variants is significantly higher than that of CombMNZ, showing improvements of between 19% and 20%.

The bpref evaluation metric also shows an average improvement for *probFuse*. However, this is attributable to the bpref score for “first” being over 30% better than that of CombMNZ, for both *probFuseAll* and *probFuseJudged*. For each of the other runs, both *probFuse* algorithms show slight performance decreases when compared to CombMNZ. These decreases never exceed 5% for either *probFuse* variant, however.

For each of the five experimental runs, the performance of *probFuseJudged* is superior to that of *probFuseAll* when evaluated using either MAP or bpref. The

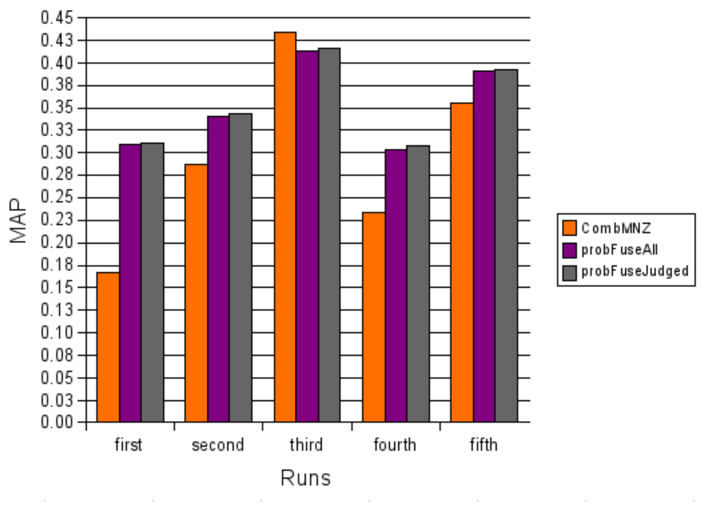

Figure 5.3: TREC-3 MAP scores for $t = 50\%$ and $x = 25$

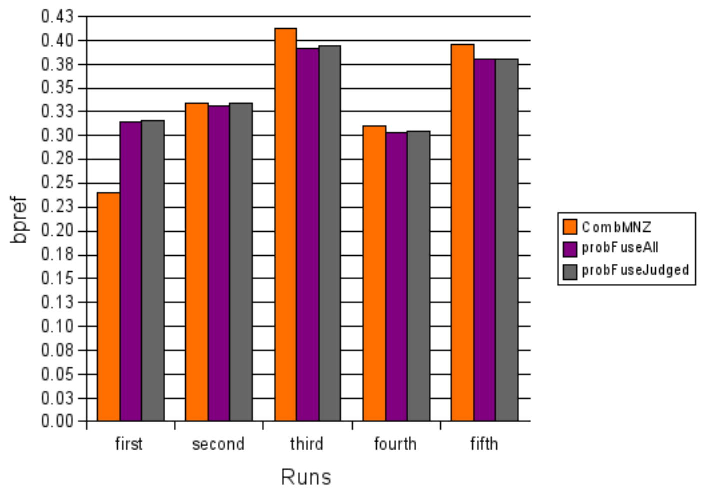

Figure 5.4: TREC-3 bpref scores for $t = 50\%$ and $x = 25$

degree to which the performance of *probFuseJudged* is higher is extremely small however. The percentage difference between the two scores only exceeds 1% for the MAP score on the “fourth” run, and is never in excess of 2%.

The data shown in Table 5.4 is illustrated in Figures 5.3 and 5.4, which show the MAP and bpref scores respectively for CombMNZ, *probFuseAll* and *probFuseJudged* when run on the TREC-3 inputs.

Table 5.5 shows similar data for the five runs that were carried out using the inputs taken from the TREC-5 data. The MAP and bpref scores for both *probFuseAll* and *probFuseJudged* are shown, along with the corresponding scores achieved by CombMNZ. This data is shown graphically in Figures 5.5 and 5.6, which show the MAP and bpref scores for each fusion technique on the TREC-

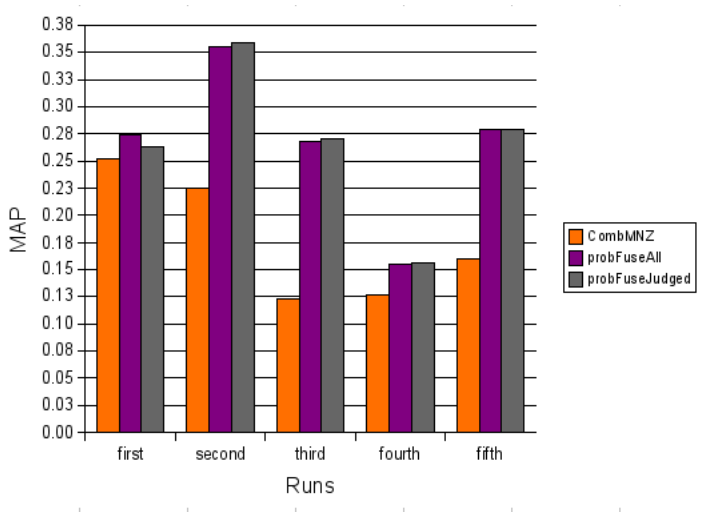

Figure 5.5: TREC-5 MAP scores for $t = 50\%$ and $x = 25$

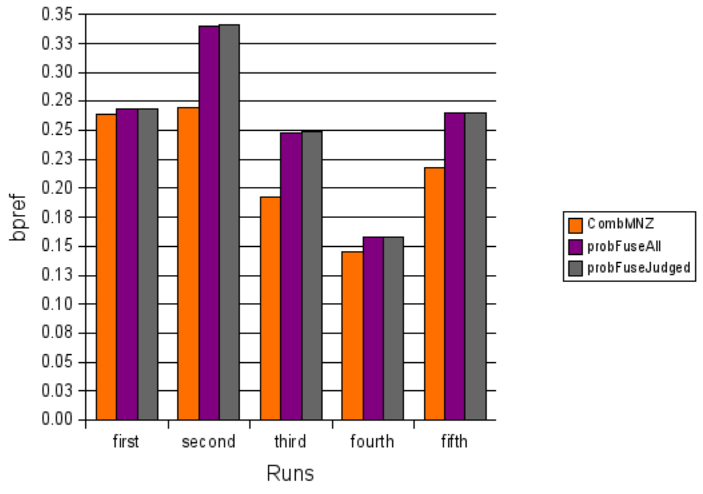

Figure 5.6: TREC-5 bpref scores for $t = 50\%$ and $x = 25$

5 dataset.

Both *probFuseAll* and *probFuseJudged* show an average increase in excess of 50% over the MAP scores achieved by CombMNZ. Additionally, both *probFuse* variations outperform CombMNZ on each of the five experimental runs. This improvement is over 50% on three runs, and only falls below 20% for the "first" run.

For the bpref measure, both *probFuseAll* and *probFuseJudged* again outperform CombMNZ, with the improvement attained being in excess of 20% for three of the five experimental runs.

The results achieved by *probFuseJudged* are again slightly better than those of *probFuseAll*, which is similar to the results of the TREC-3 experiments. Once again this improvement is very small, being less than 1% in almost all cases. The MAP score for the "first" run is the only exception and is the only case where the results of *probFuseAll* are better than those of *probFuseJudged*.

The superior performance of *probFuseJudged* is particularly interesting when analysing the bpref score. This is because, like bpref, *probFuseJudged* only takes documents that have been judged into account. For *probFuseJudged*, this behaviour occurs during the training phase. The TREC-3 and TREC-5 datasets do not have complete relevance judgements available for them, meaning that the relevance of many documents in the collection is unknown. For the level of incompleteness found in these data sets, the performance of *probFuseAll* and *probFuseJudged* is similar. Chapter 6 will investigate the effects of using a dataset for which the relevance judgments are even less complete.

5.7 Summary

This chapter outlines experiments that were carried out to demonstrate the effectiveness of the *probFuse* approach to data fusion. These were carried out using inputs from the ad hoc track of the TREC-3 and TREC-5 conferences.

Initially, the percentage of the available queries to use for training purposes was identified, along with the number of segments into which to divide each result. Separate values were not calculated for each of the input datasets. Instead, a single value for each variable that achieved high performance on both sets of inputs was selected. It was shown that by using a training set of 50% of the available queries and dividing each result set into 25 segments, *probFuse* outperformed CombMNZ on each of the two datasets. This comparison with CombMNZ was done using two evaluation metrics: Mean Average Precision and bpref. These measures were chosen so as to evaluate the overall performance of *probFuse*, as each has a very different approach to evaluating the quality of result sets. Other evaluation measures may be more appropriate to specific IR tasks. The two variations of *probFuse*, *probFuseAll* and *probFuseJudged* were both shown to outperform CombMNZ under both of these measures. In particular, the results for TREC-5 showed significant improvements over CombMNZ's MAP and bpref scores, with increases of 51% and 33% respectively.

Chapter 6 investigates the effects of using the values for t and x that were shown to perform well on the TREC-3 and TREC-5 datasets in fusion experiments on a much larger document collection.

CHAPTER

SIX

Applying *probFuse* to the TREC-2004 Web Track

6.1 Introduction

The previous chapters have demonstrated that it was possible to choose values for the training set size and the number of segments into which to divide each result set that resulted in *probFuse* outperforming CombMNZ. The key difference between this experiment and the experiments outlined in Chapter 5 is the selection of the values for t and x. In the previous experiments, these values were calculated by finding the combination of t and x values that performed best on both the TREC-3 and TREC-5 datasets. Based on these values, *probFuse* was then compared against CombMNZ. In contrast, this experiment uses these same values to perform fusion on the web track dataset from the TREC-2004 conference. Thus, it is not known if the result sets being evaluated achieve the highest performance that could be achieved with *probFuse*, as the calculation of t and x is independent of them.

The task that was set for web track participants of TREC-2004 was similar to the ad hoc task of the TREC-3 and TREC-5 conferences. The difference was that the document collection consisted of the result of a partial crawl of the .gov domain. Thus this corpus is much bigger than those used in the TREC-3 and TREC-5 ad hoc tasks, and the documents are in various formats downloaded directly from the World Wide Web (84% are HTML documents). Additionally, the increased size of the document collection means that relevance judgments are less complete, since a smaller portion of the documents in the collection have been judged for relevance to the given topics.

Section 6.2 describes the aim of this experiment, while Section 6.3 outlines the setup of the experiment. Following this, Section 6.4 discusses the results of performing fusion using *probFuse* and compares them to those of CombMNZ.

6.2 Experimental Aims

The aim of this experiment is similar to that of the previous experiments, outlined in Chapters 4 and 5. It aims to show that *probFuse* outperforms CombMNZ on a number of fusion runs.

The principal difference from the previous experiments is that both the training set size and the number of segments to divide each result set into are predetermined, rather than identifying optimal values during the experiment itself.

Additionally, the chosen document collection is much larger than those that have been used before. As a consequence, the relevance judgments are less complete than those that were used previously. This means that there are many documents in the collection that have not been judged either relevant or nonrelevant to the accompanying queries. Because *probFuse* relies on relevance judgments in its training phase, this is a factor that is likely to influence the fusion results.

6.3 Experimental Setup

The setup of the experiments is similar to the setup used for the experiments that were outlined in Chapter 5. The inputs were taken from the web track of the TREC-2004 conference [13]. Details of the data used in this chapter are given in Table 6.1. Each topfile contains 225 result sets, which represent the documents returned in response to 225 queries of different types. No distinction was made between the types of query in each case. Five fusion runs were performed, each with different sets of inputs. For each run, six input topfiles were selected at random from the 74 available. The selected topfiles are displayed in Table A.3. No input topfile was used for multiple runs.

Unlike the experiments in Chapter 5, these experiments are not concerned with identifying optimal values for t (the training set size) and x (the number of segments to divide each segment into). Instead, the same values that were calculated in Chapter 5 are used. This means that for each run, only a single value for t and x will be evaluated. The effect of this is that the t and x values used are independent of the output result sets that are produced, as they are identified during previous experiments on a different dataset. The experiments were performed using a training set size of 50% and dividing each result set into 25 segments. These are the values that were shown to perform best on the TREC-3 and TREC-5 inputs in Chapter 5.

Table 6.1: Details of the TREC-2004 web track dataset

	TREC-2004
Number of Documents	1,247,753
Collection Size	18.1Gb
Number of Participants	62
Number of Topics (Queries)	225
Average Number of Documents Judged per Topic	393.63

On each of the runs, the order of the queries was randomised five times and the three data fusion techniques, *probFuseAll*, *probFuseJudged* and CombMNZ, were applied each time.

As with the experiments in Chapter 5, the MAP and bpref measures were again used to evaluate the performance of *probFuse*. The results of this evaluation is presented in Section 6.4. Following this, further analysis of these results became necessary. Specifically, this involved examining the distribution of relevant, nonrelevant and unjudged documents in the fused result sets, the recall of *probFuse* and CombMNZ and the P10 evaluation measure. This analysis is presented in Section 6.4.1.

6.4 Evaluation and Analysis

Table 6.2 shows the MAP and bpref scores for *probFuseAll* and *probFuseJudged* for each of the five runs. Each value presented is the average of the five different orderings of the input topfiles that were used for each run. This information is presented graphically in Figures 6.1 and 6.2.

The comparison between CombMNZ and *probFuse* does not show any consistent pattern when measured using bpref. Under that measure, both *probFuseAll* and *probFuseJudged* achieve better performance than CombMNZ for two runs ("first" and "fifth"), whereas the bpref score for CombMNZ is higher on the remaining runs. The degree by which one technique achieves better performance than another varies widely. For example, on the "fourth" run,

Table 6.2: TREC-2004 performance of five individual runs for $t = 50\%$ and $x = 25$ using *probFuseAll* and *probFuseJudged*

	CombMNZ		*probFuseAll*	
	MAP	bpref	MAP	bpref
first	0.16042	0.22868	0.39154 (+144.07%)	0.29016 (+26.88%)
second	0.07808	0.31030	0.37536 (+380.74%)	0.25848 (-16.7%)
third	0.03846	0.15788	0.24418 (+534.89%)	0.13236 (-16.16%)
fourth	0.24544	0.40436	0.25862 (+5.37%)	0.14048 (-65.26%)
fifth	0.14130	0.19550	0.30278 (+114.28%)	0.21084 (+7.85%)
Average	0.13274	0.25934	0.31450 (+235.87%)	0.20646 (-12.68%)

	CombMNZ		*probFuseJudged*	
	MAP	bpref	MAP	bpref
first	0.16042	0.22868	0.39920 (+148.85%)	0.30082 (+31.55%)
second	0.07808	0.31030	0.37340 (+378.23%)	0.25558 (-17.63%)
third	0.03846	0.15788	0.24132 (+527.46%)	0.12748 (-19.26%)
fourth	0.24544	0.40436	0.25924 (+5.62%)	0.14204 (-64.87%)
fifth	0.14130	0.19550	0.30284 (+114.32%)	0.21120 (+8.03%)
Average	0.13274	0.25934	0.31520 (+234.9%)	0.20742 (-12.44%)

CombMNZ outperforms both *probFuse* techniques by over 60%, whereas for the "first" run, the bpref values for *probFuseAll* and *probFuseJudged* are both over 25% higher than the bpref score for CombMNZ.

In contrast, the MAP scores show a clear trend of *probFuse* achieving much higher performance than CombMNZ. Both *probFuse* variations achieve superior MAP scores on each of the five runs, and this increase only falls below 100% for a single run. The average improvement of *probFuse* over CombMNZ is over 230% in both cases. The only exception to these large performance increases is the "fourth" run. For that run, CombMNZ achieves its highest MAP score, as it does for bpref also. Despite this, the *probFuse* algorithms still achieve higher MAP scores, albeit to a lesser extent than the other four runs.

Comparing the performance of *probFuseAll* and *probFuseJudged*, it can be seen that neither variant significantly outperforms the other. *ProbFuseJudged* achieves slightly higher scores for both MAP and bpref on three of the five runs. This is in contrast to the experiments on the TREC-3 and TREC-5 inputs, where *probFuseJudged* achieved superior evaluation scores on almost all runs. The difference between these two approaches does not exceed 4% on any of the runs for either the MAP or bpref scores. This suggests that even for large

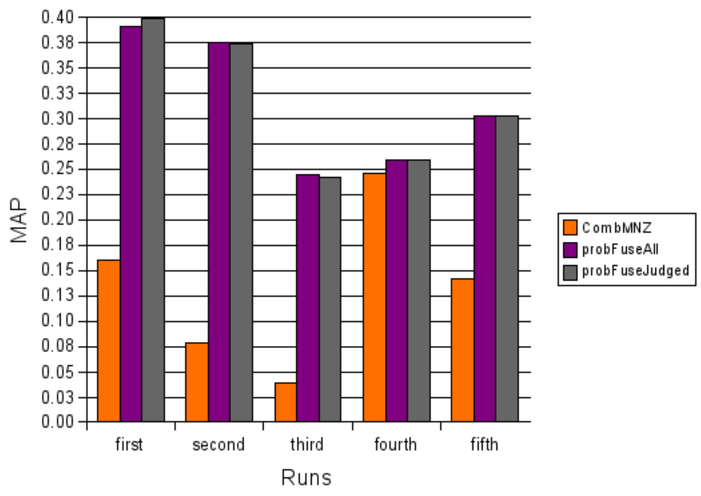

Figure 6.1: TREC-2004 MAP scores for $t = 50\%$ and $x = 25$

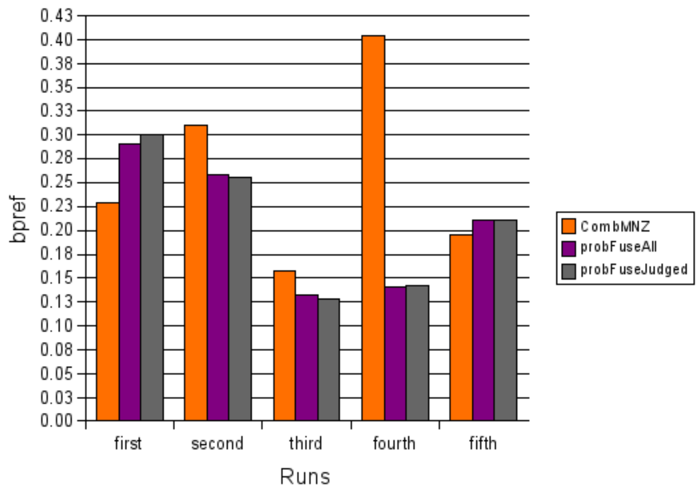

Figure 6.2: TREC-2004 bpref scores for $t = 50\%$ and $x = 25$

document collections with limited relevance judgments, the decision to base the probability calculation for *probFuse* on all documents or judged documents does not have a significant effect on fusion performance.

The fact that *probFuse* achieves a lower average bpref score than CombMNZ, despite significant increases in MAP scores, is a cause for some concern. These scores are examined in more detail in Section 6.4.1, with reference to the distribution of judged relevant, judged nonrelevant and unjudged documents in the result sets returned by each of the fusion algorithms.

6.4.1 Analysis of bpref scores

This section examines the distribution of judged relevant, judged nonrelevant and unjudged documents in the result sets produced by *probFuseAll*, *probFuse-Judged* and CombMNZ. The motivation for this is to explain the differences between the evaluation results produced using MAP and those using bpref. Using the bpref measure, CombMNZ achieved a higher level of performance than *probFuse* on average, although it did not achieve superior results on all five runs. In contrast, MAP showed *probFuse* to outperform CombMNZ to a significant degree, with increases of over 100% in four out of five runs.

Figures 6.3, 6.4 and 6.5 show the distribution of judged relevant, judged nonrelevant and unjudged documents respectively. Each of these figures contains a line representing each of CombMNZ, *probFuseAll* and *probFuseJudged*. These lines represent the average distribution over all the result sets produced by the relevant fusion technique for this experiment. This is the average for all five fusion runs. Each data point represents the percentage of documents in a particular position in the result sets that are judged relevant, judged nonrelevant or unjudged. The y-axis shows this percentage, while the x-axis shows the position in the result set. Each point along the x-axis represents a group of ten documents. For example, data points for position 0 in Figure 6.3 on the x-axis represents the percentage of documents returned in positions 0 to 9 in the fused result sets were judged relevant. Similarly, at 1 on the x-axis, it is the documents returned in positions 10 to 19 that are considered. In this dataset, there are far fewer judged relevant documents than judged nonrelevant or unjudged documents. For this reason, the percentages being shown for the judged relevant document distribution are much smaller than for the judged nonrelevant and unjudged distributions. The scale of Figure 6.3 is different from that of Figures 6.4 or 6.5 so that the data can be viewed more easily.

The two *probFuse* variants return more relevant documents than CombMNZ at the beginning of their fused result sets. This will have a positive effect on MAP, as the precision of these documents appearing at or near the top of the result set will be high. A much greater number of early documents returned by CombMNZ are unjudged. This has a detrimental effect on its MAP scores, as these will effectively be considered to be nonrelevant. However, this will not affect bpref, despite the fact that fewer relevant documents are being returned in early positions. For both *probFuseAll* and *probFuseJudged*, more judged nonrelevant documents are being returned at the top of the result sets

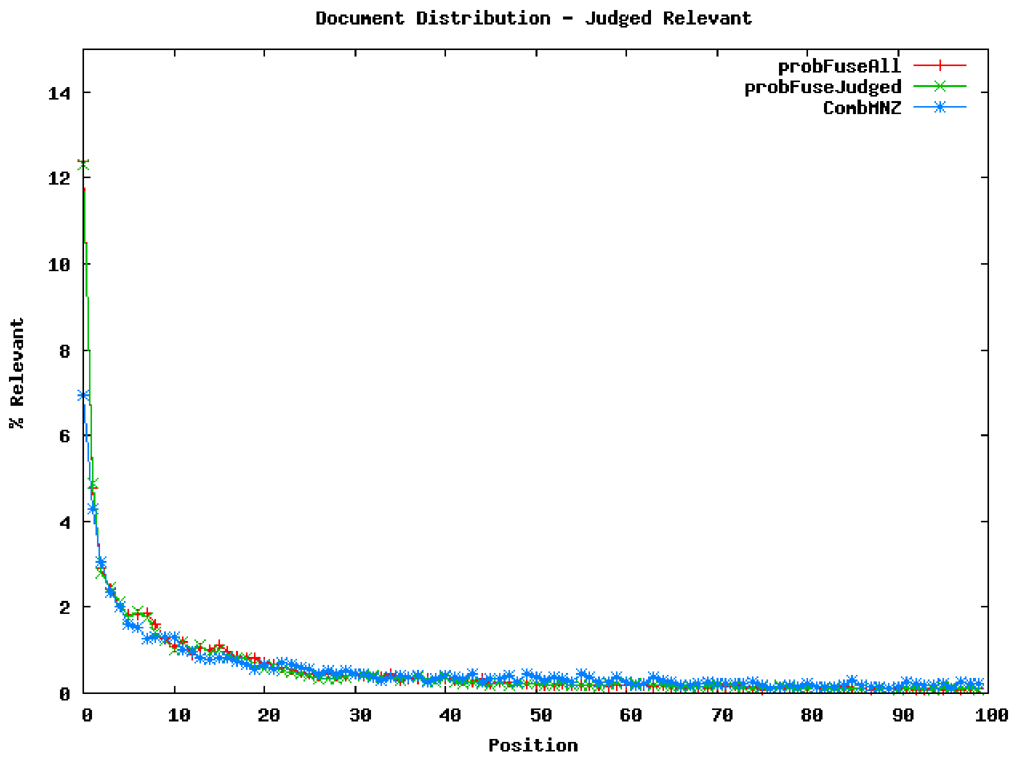

Figure 6.3: Distribution of judged relevant documents for three fusion algorithms

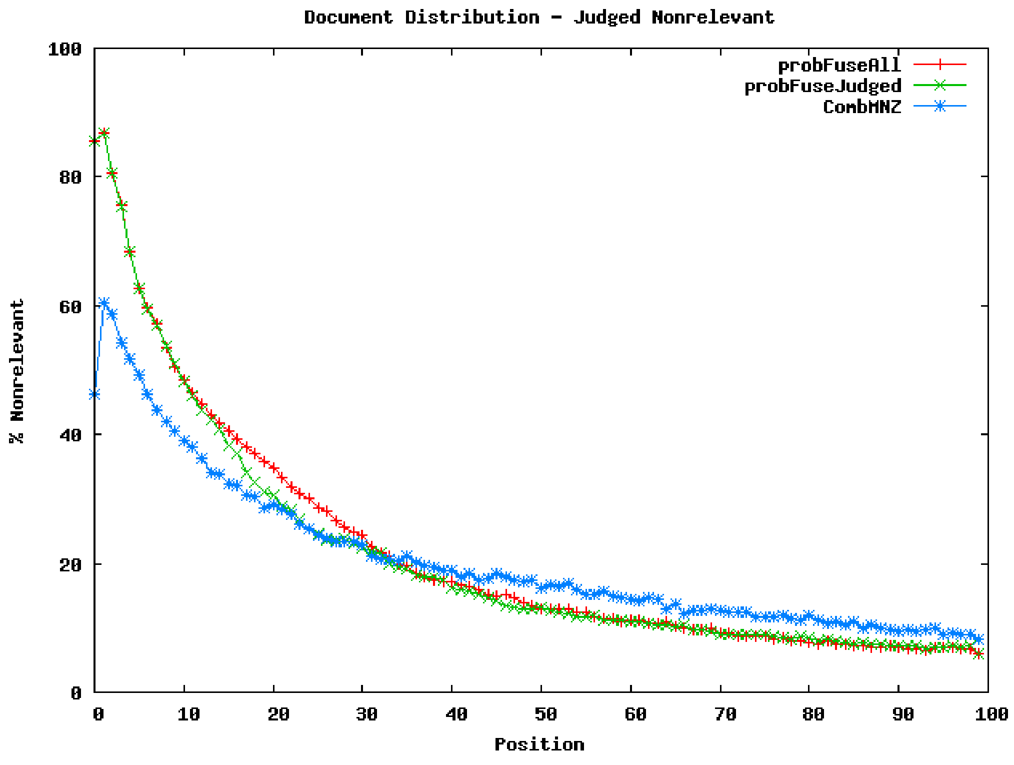

Figure 6.4: Distribution of judged nonrelevant documents for three fusion algorithms

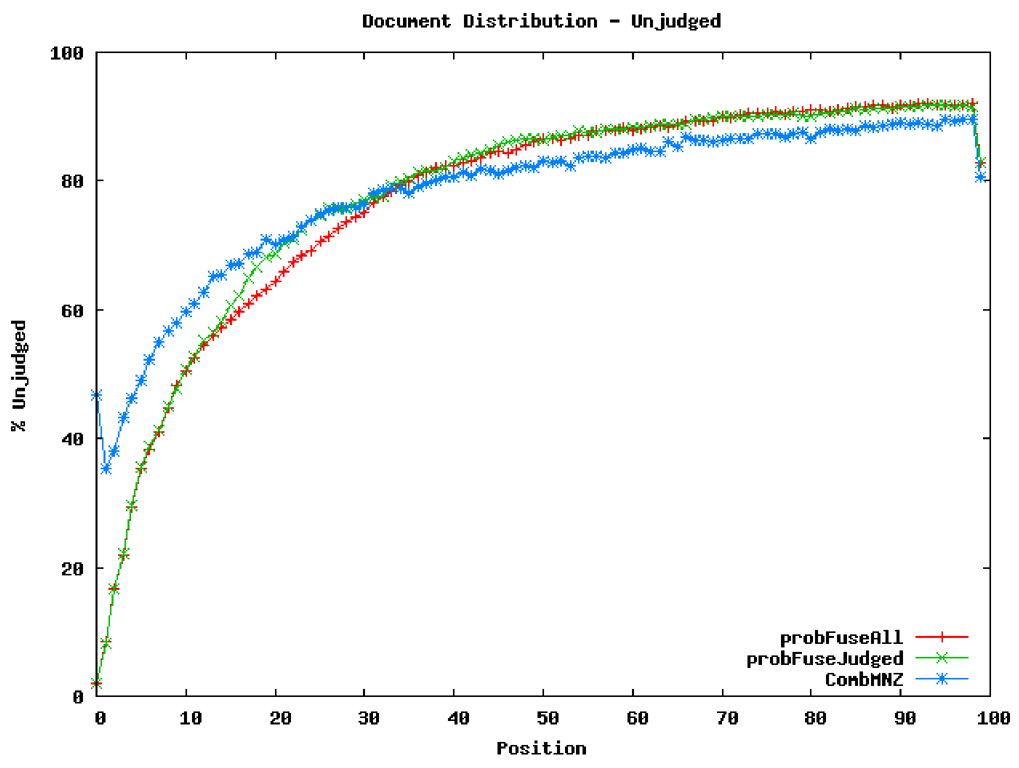

Figure 6.5: Distribution of unjudged documents for three fusion algorithms

than CombMNZ. The effect of this on MAP will be no different to returning unjudged documents in similar positions. This does, however, have a detrimental effect on the bpref scores *probFuse* achieves.

In addition to returning fewer relevant documents in early positions in the fused result sets it produces, Table 6.3 shows that overall, CombMNZ also displays lower recall. Recall is described in Section 2.3.1. This table shows the average number of relevant documents contained in the topfiles returned by each of the three data fusion techniques over each of the five runs. The "Average Relevant Documents" shows that there was an average of 882.68 judged relevant documents available for retrieval over all the queries that were used for fusion. The reason this is an average is because the queries used for fusion varied as the order of the queries was randomised, as only 50% of the queries were used for fusion. Using 50% of the available queries for training purposes resulted in 113 queries being used for fusion in each case. This means that there is an average of only 7.81 judged relevant documents for each query. This is another issue which can be taken into account when interpreting the bpref scores attributed to each fusion technique.

For a query with R judged relevant documents, the bpref measure only takes

Table 6.3: Average number of relevant documents returned

Average Relevant Documents	882.68
probFuseAll	690.84 (78.27%)
probFuseJudged	670.98 (76.01%)
CombMNZ	661.96 (74.99%)

the first *R* nonrelevant documents returned into account. For example, consider a query that has 8 relevant documents and for which at least 8 nonrelevant documents have been returned in the first 20 positions in a result set. In these circumstances, a relevant document returned in 21st position would have the same effect on bpref score as a relevant document returned in the 1000th position or not returned at all. In contrast, the effect of a relevant document on the MAP score continually declines as its position increases. Under MAP, a relevant document that is returned in the result set will always be considered to be preferable to one that is not returned at all.

The tendency of *probFuse* to return more relevant documents in early positions also motivates the introduction of the P10 evaluation metric. This measures the precision of a result set after ten documents have been returned. Previous research has indicated that these are the positions in which typical users expect to find relevant documents. One study has found that 85.2% of users examine only the top 10 documents or fewer [51]. As such, P10 is an important measure that is particularly suited to the web retrieval task.

The results of the evaluation of *probFuse* and CombMNZ using the P10 metric are presented in Table 6.4 and Figure 6.6. From these, it can be seen that both variations of *probFuse* achieve higher performance than CombMNZ on all five runs. In particular, the "second", "third" and "fifth" runs show a substantial improvement in the P10 score. As with the MAP and bpref measures, the two variations of *probFuse* achieve similar performance to each other. This data indicates that *probFuse* performs better at retrieving documents in the positions most likely to be examined by users of an IR system.

Overall, CombMNZ returned more unjudged documents in early positions in its fused result sets. This has the effect of allowing judged relevant documents to be returned later in the result set, without adversely affecting the bpref score, although it is reflected in the poor MAP scores achieved. The MAP and P10 measures have rewarded *probFuseAll* and *probFuseJudged* for returning relevant documents in early positions. This is particularly important given that most users do not examine documents outside the top ten results. Although

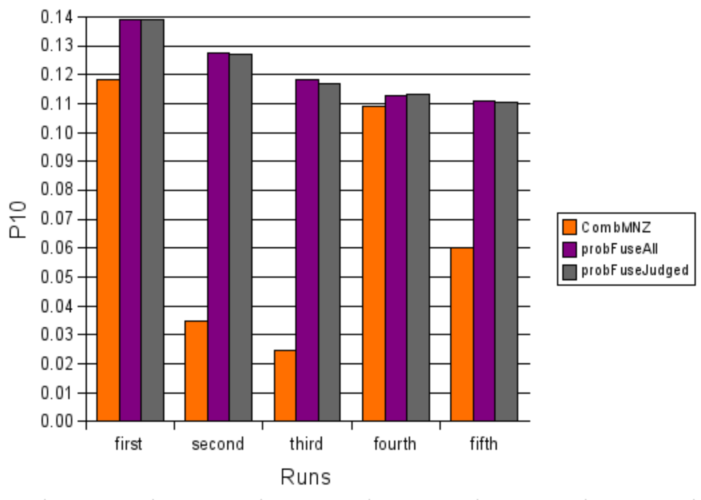

Figure 6.6: TREC-2004 P10 scores for $t = 50\%$ and $x = 25$

Table 6.4: TREC-2004 performance of five individual runs for $t = 50\%$ and $x = 25$ using *probFuseAll* and *probFuseJudged*, evaluated with P10

	CombMNZ	*probFuseAll*	*probFuseJudged*
first	0.11858	0.13930 (+17.47%)	0.13910 (+17.31%)
second	0.03486	0.12780 (+266.61%)	0.12708 (+264.54%)
third	0.02480	0.11824 (+378.77%)	0.11700 (+371.78%)
fourth	0.10902	0.11292 (+3.58%)	0.11310 (+3.74%)
fifth	0.06038	0.11116 (+84.1%)	0.11046 (+82.94%)
Average	0.06953	0.12188 (+149.71%)	0.12135 (+148.06%)

bpref has not penalised CombMNZ for returning relevant documents further down the fused result sets, such performance is unlikely to be beneficial in a real-world situation where the relevance of the top-ranked documents is of utmost importance.

6.5 Summary

In contrast with the experiments described in Chapters 4 and 5, this chapter describes an experiment to demonstrate the effect of using predetermined Values for the training set size and the number of segments into which to divide each result set. These values were those that were shown to achieve optimal performance on the TREC-3 and TREC-5 datasets in Chapter 5. The difference between this experiment and the previous experiments outlined in Chapters 4 and 5 is that in this case, these values are not optimised for the dataset in question, but are taken from other experiments on independent datasets.

In running this experiment, data from the web track of the TREC-2004 conference was used. As the document collection being queried for the web track task is far larger than those used for TREC-3 or TREC-5, the relevance judgments for this dataset are far less complete than those for any collection used in the previous experiments.

The experiment showed that less complete relevance judgements did not cause *probFuseAll* and *probFuseJudged* to perform significantly differently. The MAP evaluation measure showed both *probFuse* variants to significantly outperform CombMNZ. The bpref measure did not show one technique to conclusively outperform the others. Although the average bpref score for CombMNZ was higher, it only achieved higher scores in three out of five runs. However, *probFuse* was shown to have greater overall recall, and demonstrated a greater tendency to return relevant documents in early positions in its fused result sets. This latter tendency is reflected in an average improvement in the P10 evaluation measure of over 148%. These facts, combined with the average increase in MAP score of over 230% justifies the conclusion that *probFuse* achieves significantly superior results to CombMNZ.

CHAPTER

SEVEN

Conclusions and Future Work

This thesis presents *probFuse*, a probabilistic approach to the task of data fusion. *ProbFuse* assumes that the results produced by an IR system on a number of training queries is indicative of its future performance. Documents in the fused result set are ranked according to the probability that they are relevant to the given query. This approach has been shown to achieve promising results when evaluated on a number of document collections. This chapter proposes a number of possible directions for future work and summarises the conclusions drawn as a result of the work presented in this thesis.

7.1 Future Work

All of the experiments presented in this thesis have been run on static test collections for which relevance judgments are available. These relevance judgments can be used to calculate a document's probability of relevance. However, in a large-scale, dynamic IR system (such as one designed to retrieve documents from the World Wide Web), these relevance judgments are unavailable. For this reason, it is important to develop an alternative means of constructing the probability of relevance distribution. The absence of relevance judgments also makes it difficult to evaluate the performance of the fusion algorithm.

As a possible method of calculating the probability of relevance for document collections for which relevance judgments are unavailable, it is necessary to investigate the effects of training *probFuse* on one document collection and using this training data to perform fusion on another collection. This is not possible with the current TREC data, as the systems that have been used as inputs are not the same for each conference. To date, the only data that has taken from one collection and applied to experiments run on another are the values for t and

x that were used for the experiments in Chapter 6. Performing experiments on this will require result sets for multiple document collections produced by the same input systems.

The effect of different levels of relevance judgment completeness is another factor worthy of further investigation. One approach to this would be to vary the number of relevance judgments available for *probFuse*'s training phase, while maintaining the same level of completeness for evaluating the fusion output.

Result sets taken from TREC submissions are limited to 1000 documents in length. Constant-length result sets do not commonly occur in web-based IR systems. Different queries are likely to have different numbers of relevant documents associated with them. Additionally, IR algorithms will have different methods of estimating relevance and so they are likely to produce result sets of different lengths to each other. For this reason, it is important to investigate the effect of varying the length of the input result sets.

To date, *probFuse* uses a trained, a priori, probability to weight each input system. Other approaches to estimating the probability of relevance may also be possible, without the necessity of an initial training phase. One approach may be to build on work done by Manmatha et al. [34], who proposed a method of mapping the scores attributed to documents by IR systems to probabilities of relevance. An alternative approach may be to adjust the probabilities associated with each input system at query time, according to user behaviour.

7.2 Conclusions

The *probFuse* algorithm is described in Chapter 3. This included the proposal of two variants of probability calculations, *probFuseAll* and *probFuseJudged*. These allow for the investigation of the effects of using all documents in a result set for training purposes or using just documents for which relevance judgments are available. The *probFuse* algorithm includes two variables: t (the percentage of available queries to be used for training purposes) and x (the number of segments into which to divide each result set). Optimal values for these must be determined empirically.

An exploratory study was initially carried out on a number of small document collections in order to test the effectiveness of *probFuse*. This is described in Chapter 4. These collections were chosen as complete relevance judgements are available for them. This means that for every document in the collection, it is known whether it is relevant or nonrelevant to all the accompanying queries. This study showed that *probFuse* outperformed each of the individual inputs and also outperformed CombMNZ. These results prompted further investigation of the performance of *probFuse* on larger document collections.

Progressing from this initial study, Chapter 5 made use of inputs from the larger TREC-3 and TREC-5 datasets. A feature of these document collections is that the relevance judgments available are incomplete, meaning that the relevance of many documents is unknown. The aim of these experiments was to demonstrate that values for t and x could be found that would cause *probFuse* to outperform the CombMNZ algorithm on both collections. It was observed that by using a training set size of 50% and dividing each result set into 25 segments, this goal was achieved.

The final experiment presented in this thesis was to investigate whether it was possible to use the values for t and x that were found to perform optimally on the TREC-3 and TREC-5 datasets and use them to perform fusion on a different set of inputs on a larger document collection. This experiment is described in Chapter 6. The inputs were taken from the web track of the TREC-2004 conference. This data includes relevance judgments that are far less complete than any of the datasets that had previously been used. Despite the small number of available relevance judgments, no significant difference between the performance of the two variations of the *probFuse* probability calculation was observed. However, the difference between the performance of *probFuse* and CombMNZ was observed to be substantial. It was shown that:

- *probFuse* achieved increases of over 230% when evaluated using MAP.
- *probFuse* achieved greater recall overall, by returning more relevant documents than CombMNZ
- *probFuse* showed a greater tendency to return relevant documents in early positions, which is where users are most likely to look for documents that satisfy their information need. This is reflected in an average improvement of over 148% when evaluated using the P10 measure.

The other evaluation measure used, bpref, failed to conclusively show one fusion technique having superior performance over the other. This is discussed in 6.4.1.

Overall, *probFuse* has been shown to outperform CombMNZ on all types of document collections. This includes small collections such as Cranfield and LISA, the larger collections from TREC-3 and TREC-5 and also the TREC web track. As the size of the document collections has increased, the completeness of the available relevance judgments has declined, without having an adverse effect on the performance of *probFuse*.

The results of the web track experiment are particularly promising, as they show that *probFuse* has the potential to be applied to large document collections such as in the web search domain, where it is necessary to infer relevance and judgments will be extremely incomplete.

Corporate intranets represent a target that will be more achievable in the short term. These contain far fewer documents than are found on the World Wide Web and are comparable in size to the larger test collections used in the experiments presented in this thesis. A benefit of this smaller size is that it the production of relevance judgments becomes far less challenging. Given these relevance judgments, *probFuse* has demonstrated that it is capable of achieving high performance data fusion in situations such as this.

APPENDIX

A

Inputs to the TREC Fusion Experiments

This appendix lists the topfiles that were used for the fusion experiments presented in Chapters 5 and 6. These topfiles were selected at random from all the available topfiles for the relevant TREC conferences.

Table A.1: Inputs to TREC-3 ad hoc experimental runs

first	second	third	fourth	fifth
acqnt1	clartm	brkly7	assctv1	assctv2
citri1	crnlla	clarta	erima1	nyuir1
crnlea	dortd2	dortd1	lsia0mf	rutfua1
padre2	eth002	eth001	lsia0mw2	rutfua2
xerox3	nyuir2	inq101	virtu1	siems1
xerox4	padre1	pircs1	vtc2s2	westp1

Table A.2: Inputs to TREC-5 ad hoc experimental runs

first	second	third	fourth	fifth
brkly18	anu5man4	anu5aut2	DCU962	anu5aut1
DCU963	CLCLUS	city96a1	genrl1	colm4
ETHal1	erliA1	CLTHES	ibmge1	Cor5A2cr
KUSG3	genrl3	ETHas1	ibms96a	LNmFull1
vtwnA1	ibms96b	genrl4	KUSG2	LNmFull2
vtwnB1	uwgcx0	ibmgd2	mds003	pircsAAL

Table A.3: Inputs to the TREC-2004 web track experimental runs

first	second	third
csiroatnist	ICT04CIIS1AT	humW04dp
fdwiedf0	MeijiHILw5	humW04dpl
mpi04web01	MU04web1	MSRAmixed1
MSRC04B1S2	MU04web3	MSRAx5
MSRC04B3S	MU04web4	MU04web5
MU04web2	uogWebSelAnL	uogWebCA

fourth	fifth
MeijiHILw1	fdwiellq0
MeijiHILw3	fdwiesl0
MSRAmixed3	humW04pl
THUIRmix043	MeijiHILw2
UAmsT04MWScb	mpi04web02
VTOK5	mpi04web08

APPENDIX

B

Results of Individual Runs on TREC-3

This appendix shows the results of each of the five fusion runs that were run on the TREC-3 dataset in Chapter 5. Each figure shows the MAP scores achieved for each value of x when using a training set size of 50%. The average of the MAP scores in each of these figures is displayed in Figure 5.1.

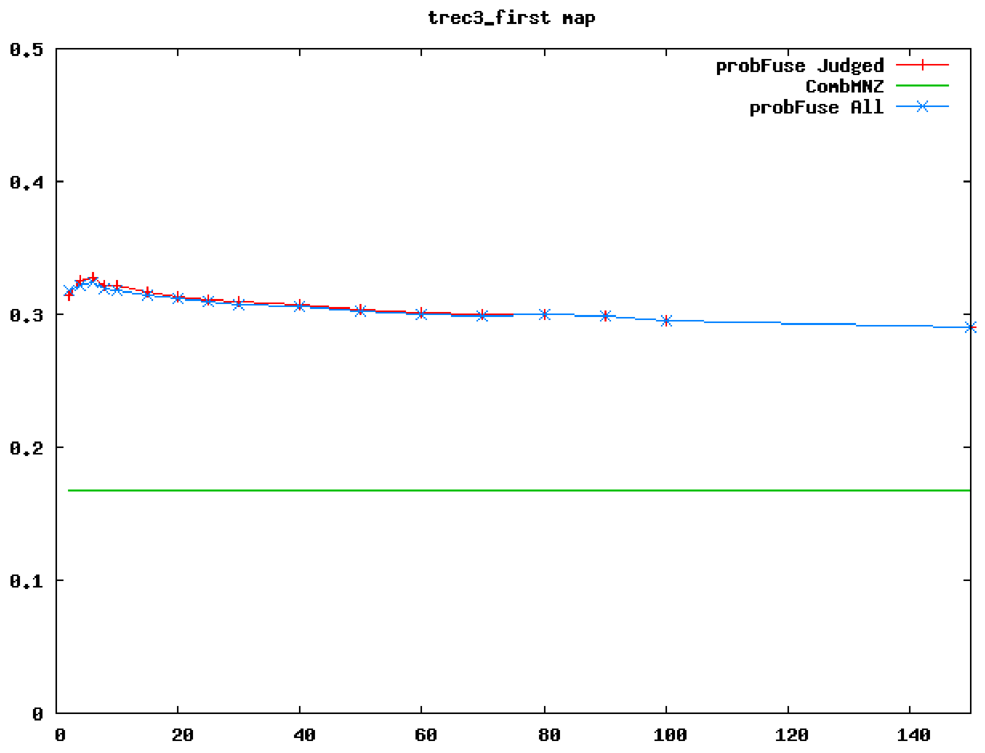

Figure B.1: TREC-3 MAP scores for $t = 50\%$ for the "first" run

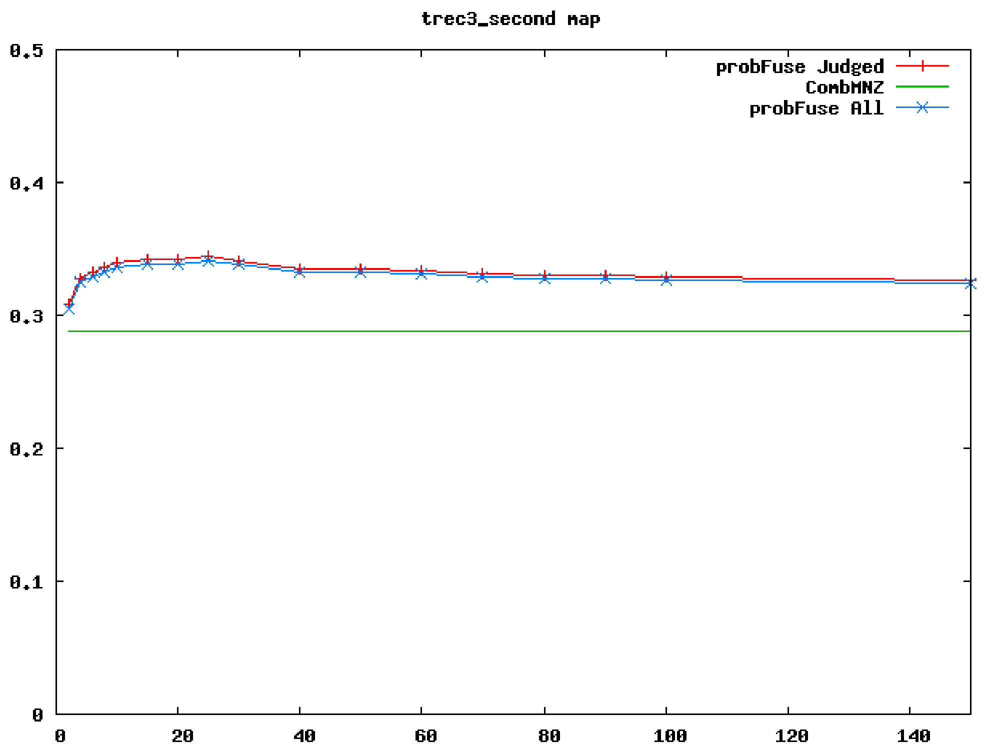

Figure B.2: TREC-3 MAP scores for $t = 50\%$ for the "second" run

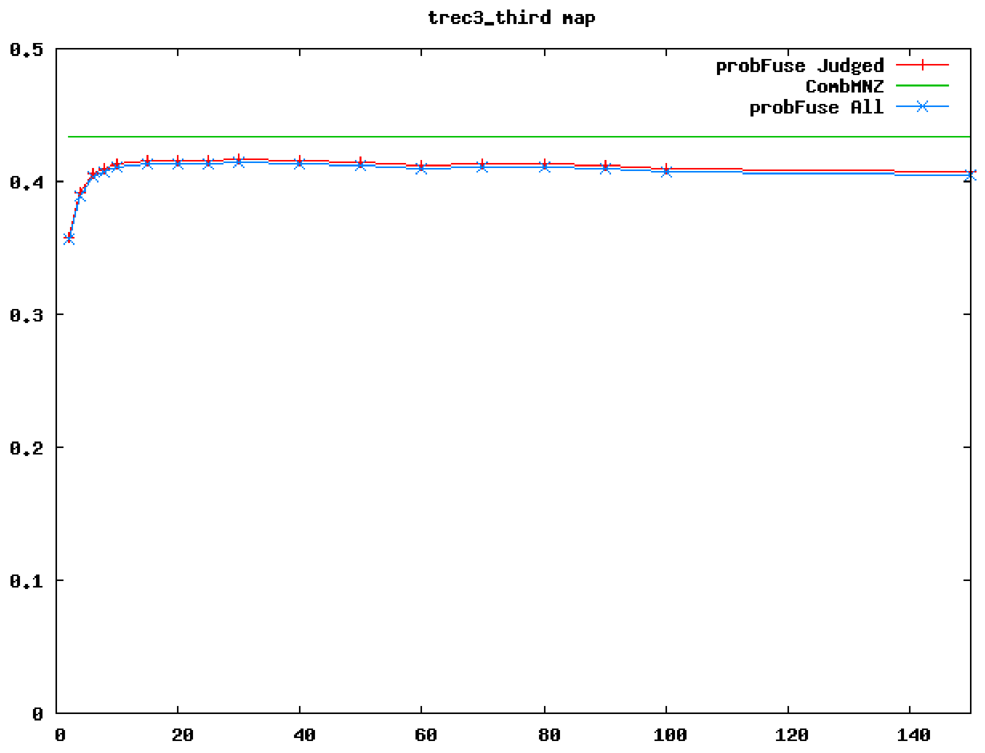

Figure B.3: TREC-3 MAP scores for $t = 50\%$ for the "third" run

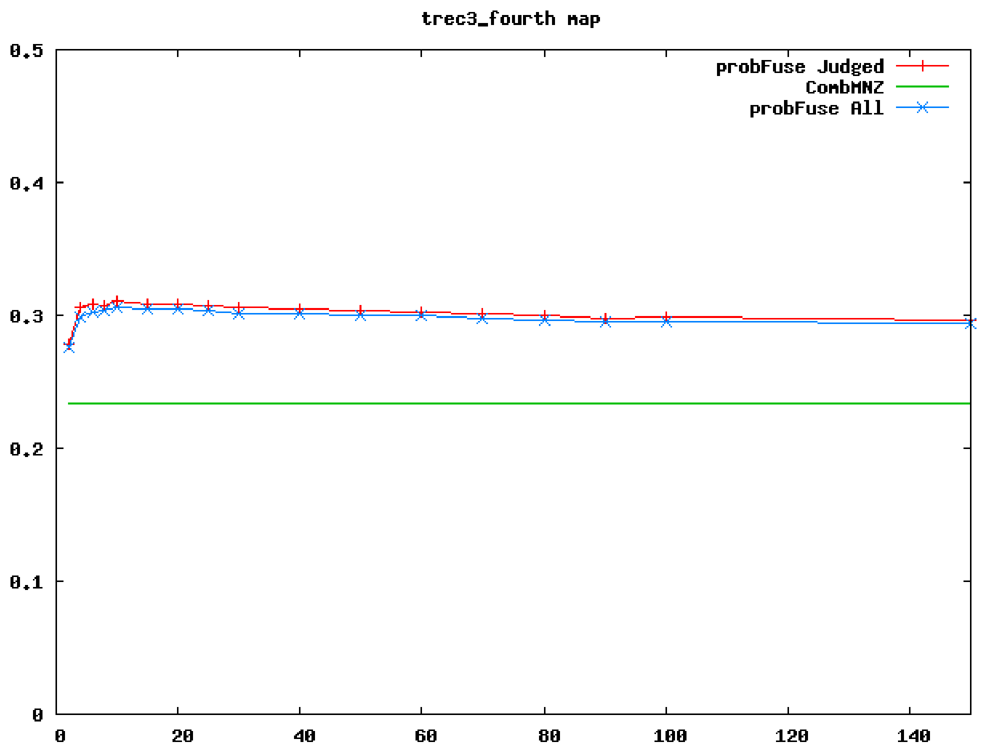

Figure B.4: TREC-3 MAP scores for $t = 50\%$ for the "fourth" run

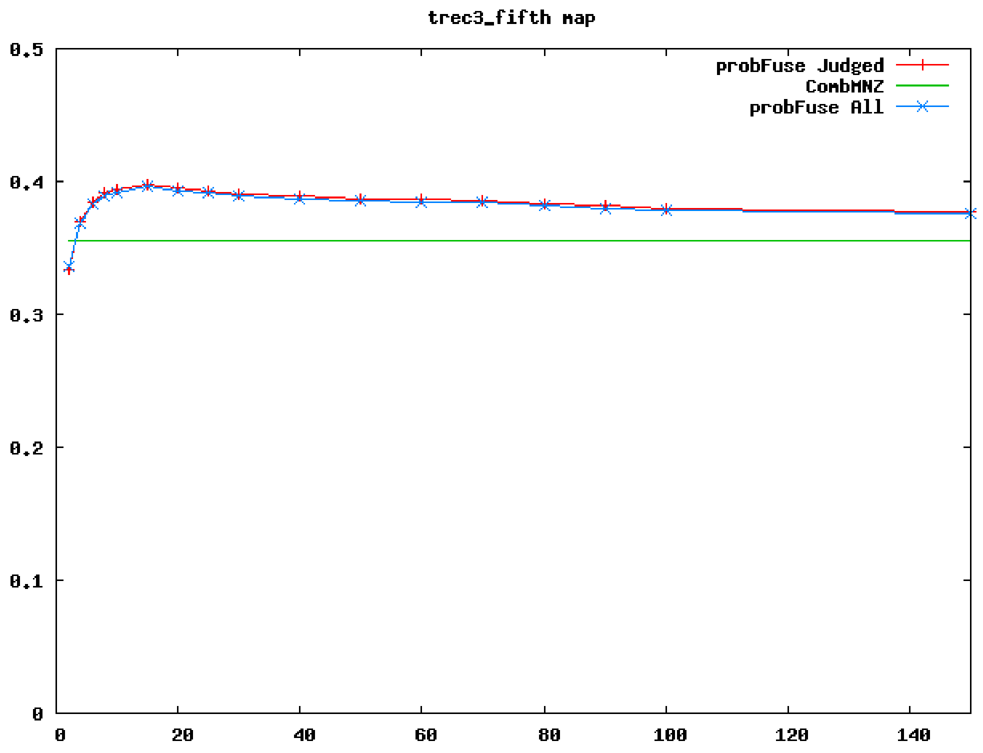

Figure B.5: TREC-3 MAP scores for $t = 50\%$ for the "fifth" run

APPENDIX

C

Results of Individual Runs on TREC-5

This appendix shows the results of each of the five fusion runs that were run on the TREC-5 dataset in Chapter 5. Each figure shows the MAP scores achieved for each value of x when using a training set size of 50%. The average of the MAP scores in each of these figures is displayed in Figure 5.2.

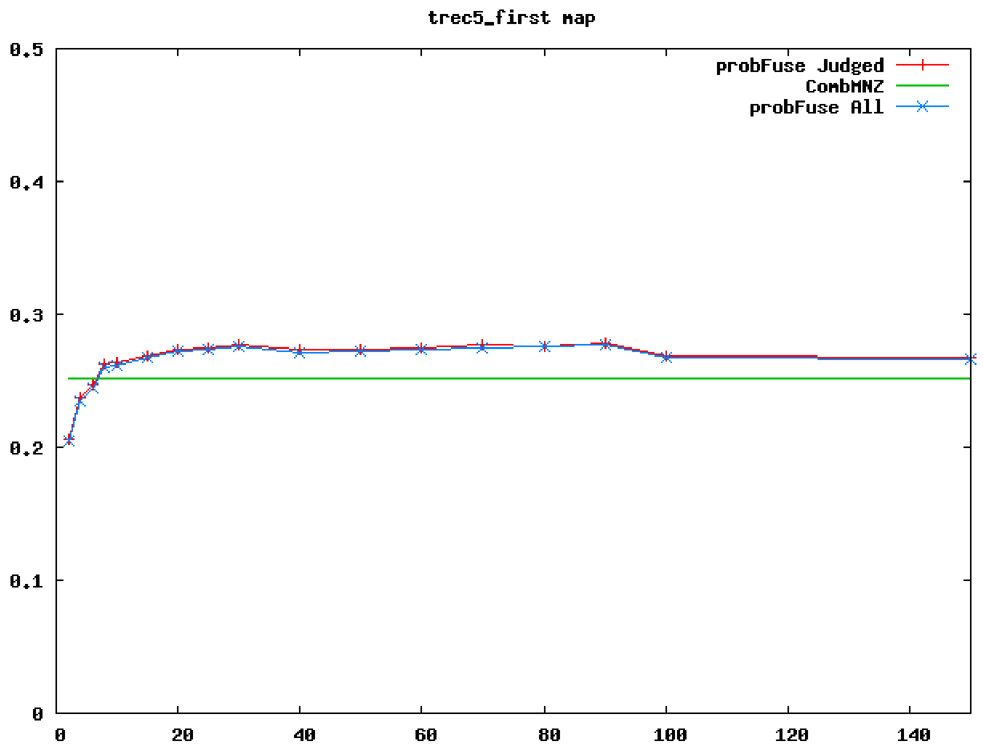

Figure C.1: TREC-5 MAP scores for $t = 50\%$ for the "first" run

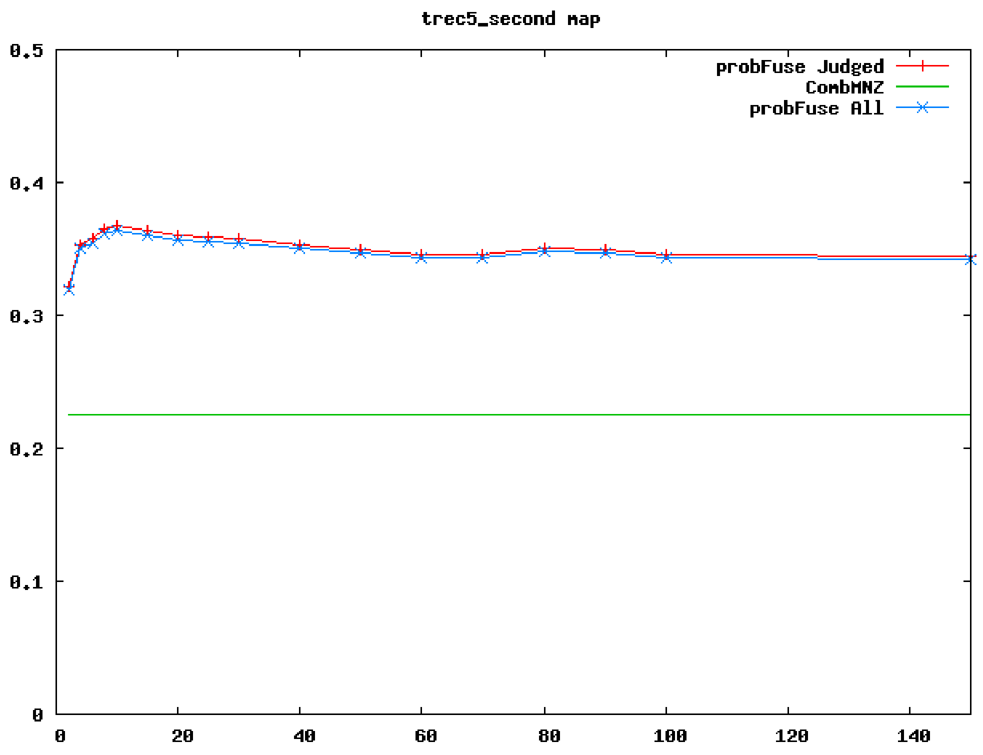

Figure C.2: TREC-5 MAP scores for $t = 50\%$ for the "second" run

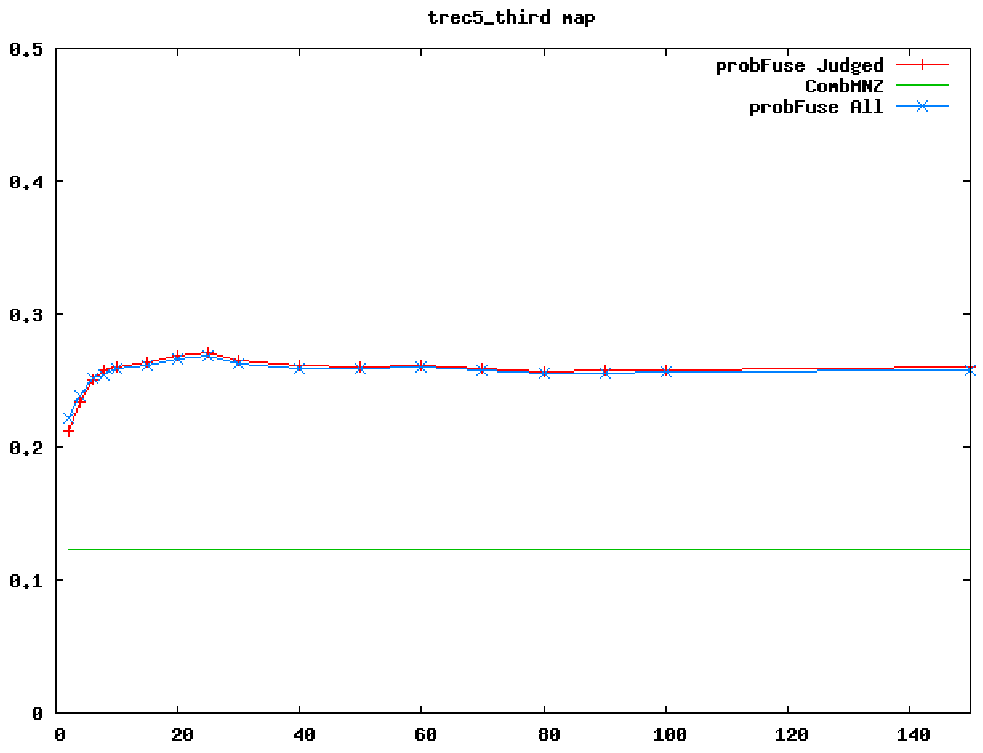

Figure C.3: TREC-5 MAP scores for $t = 50\%$ for the "third" run

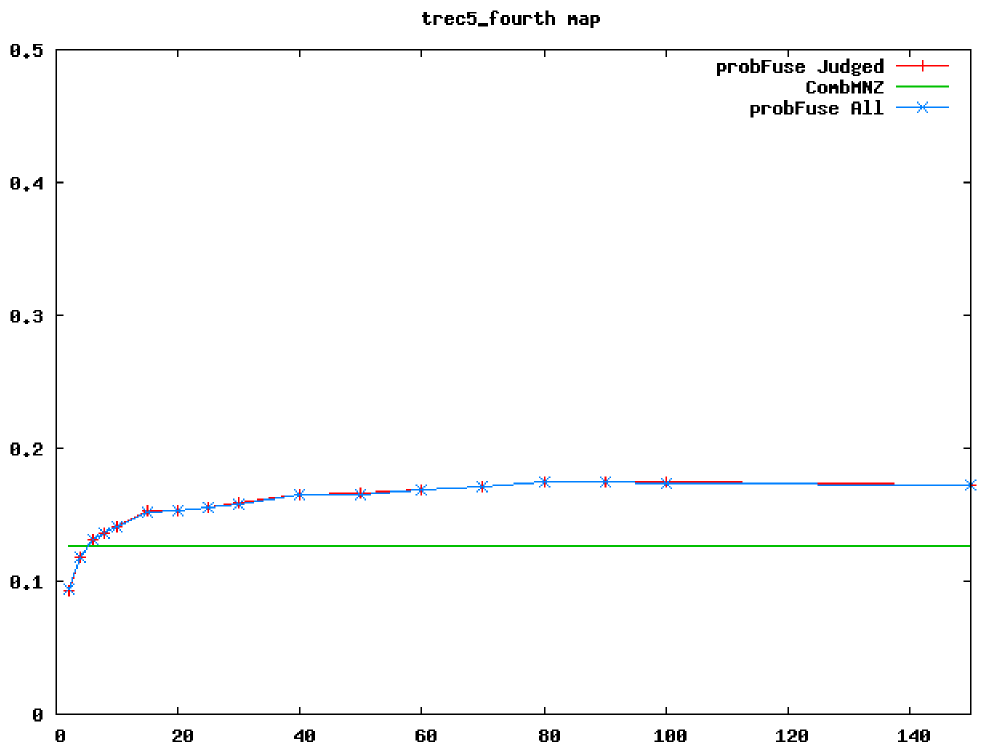

Figure C.4: TREC-5 MAP scores for $t = 50\%$ for the "fourth" run

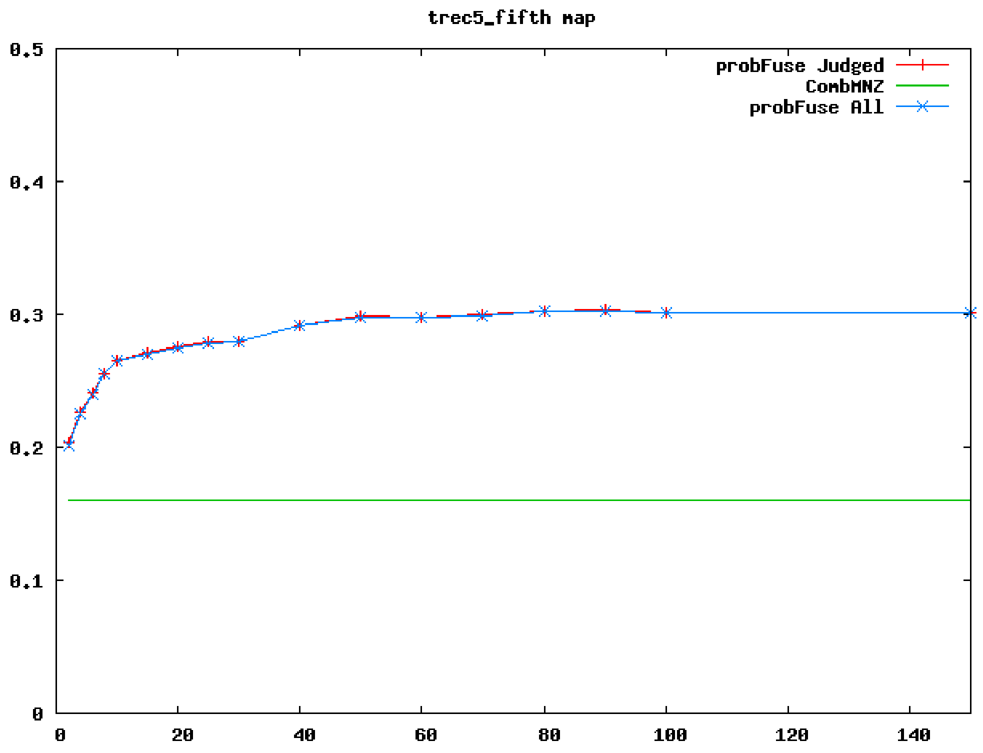

Figure C.5: TREC-5 MAP scores for $t = 50\%$ for the "fifth" run

BIBLIOGRAPHY

[1] Javed A. Aslam and Mark Montague. Bayes optimal metasearch: a probabilistic model for combining the results of multiple retrieval systems. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 379–381, New York, NY, USA, 2000. ACM Press.

[2] Javed A. Aslam and Mark Montague. Models for metasearch. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 276–284, New York, NY, USA, 2001. ACM Press.

[3] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[4] Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Automatic combination of multiple ranked retrieval systems. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 173–181, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[5] Steven M. Beitzel, Ophir Frieder, Eric C. Jensen, David Grossman, Abdur Chowdhury, and Nazli Goharian. Disproving the fusion hypothesis: an analysis of data fusion via effective information retrieval strategies. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 823–827, New York, NY, USA, 2003. ACM Press.

[6] Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, David Grossman, Ophir Frieder, and Nazli Goharian. Fusion of effective retrieval strategies in the same information retrieval system. *J. Am. Soc. Inf. Sci. Technol.*, 55(10):859–868, 2004.

[7] Nicholas J. Belkin, C. Cool, W. Bruce Croft, and James P. Callan. The effect of multiple query representations on information retrieval system performance. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346, New York, NY, USA, 1993. ACM Press.

[8] Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32, New York, NY, USA, 2004. ACM Press.

[9] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–28, New York, NY, USA, 1995. ACM Press.

[10] Jamie Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 479–490, New York, NY, USA, 1999. ACM Press.

[11] Abdur Chowdhury, Ophir Frieder, David Grossman, and Catherine McCabe. Analyses of multiple-evidence combinations for retrieval strategies. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 394–395, New York, NY, USA, 2001. ACM Press.

[12] Philipp Cimiano and Steffen Staab. Learning by googling. *SIGKDD Explor. Newsl.*, 6(2):24–33, 2004.

[13] Nick Craswell and David Hawking. Overview of the TREC-2004 web track. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC-2004)*, 2004.

[14] Nick Craswell, David Hawking, and Paul B. Thistlewaite. Merging results from isolated search engines. In *Australasian Database Conference*, pages 189–200, Auckland, New Zealand, 1999.

[15] Padima Das-Gupta and Jeffrey Katzer. A study of the overlap among document representations. In *SIGIR '83: Proceedings of the 6th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 106–114, New York, NY, USA, 1983. ACM Press.

[16] Owen de Kretser, Alistair Moffat, Tim Shimmin, and Justin Zobel. Methodologies for distributed information retrieval. In *ICDCS '98: Proceedings of the The 18th International Conference on Distributed Computing Systems*, page 66, Washington, DC, USA, 1998. IEEE Computer Society.

[17] Jay L. Devore and Roxy Peck. *Statistics: the exploration and analysis of data*. Brooks/Cole Publishing Company, 3rd edition, 1997.

[18] Edward A. Fox and Joseph A. Shaw. Combination of multiple searches. In *Proceedings of the 2nd Text REtrieval Conference (TREC-2), National Institute of Standards and Technology Special Publication 500-215*, pages 243–252, 1994.

[19] Susan Gauch, Guijun Wang, and Mario Gomez. Profusion: Intelligent fusion from multiple, distributed search engines. *Journal of Universal Computer Science*, 2(9):637–649, 1996.

[20] Luis Gravano, Kevin Chang, Hector Garcia-Molina, and Andreas Paepcke. Starts: Stanford protocol proposal for internet retrieval and search. Technical report, Stanford, CA, USA, 1997.

[21] Luis Gravano and Hector Garcia-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In *VLDB '95: Proceedings of the 21st International Conference on Very Large Data Bases*, pages 78–89, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[22] Venkat N. Gudivada, Vijay V. Raghavan, William I. Grosky, and Rajesh Kasanagottu. Information retrieval on the world wide web. *IEEE Internet Computing*, 01(5):58–68, 1997.

[23] Donna Harman. Overview of the first Text REtrieval Conference (TREC-1). In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 36–47, New York, NY, USA, 1993. ACM Press.

[24] Donna Harman. Overview of the third Text REtrieval Conference (TREC-3). In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 1–19, 1994.

[25] David Hawking and Paul Thistlewaite. Methods for information server selection. *ACM Trans. Inf. Syst.*, 17(1):40–76, 1999.

[26] Adele E. Howe and Daniel Dreilinger. SavvySearch: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.

[27] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[28] Karen Sparck Jones and Peter Willett, editors. *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[29] Leah S. Larkey, Margaret E. Connell, and Jamie Callan. Collection selection and results merging with topically organized U.S. patents and TREC data. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 282–289, New York, NY, USA, 2000. ACM Press.

[30] S. Lawrence and C.L. Giles. Searching the world wide web. *Science*, 280(5360):98–100, 1998.

[31] Steve Lawrence and C. Lee Giles. Inquirus, the NECI meta search engine. In *Seventh International World Wide Web Conference*, pages 95–105, Brisbane, Australia, 1998. Elsevier Science.

[32] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Intelligence*, 11(1):32–39, 2000.

[33] Joon Ho Lee. Analyses of multiple evidence combination. *SIGIR Forum*, 31(SI):267–276, 1997.

[34] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–275, New York, NY, USA, 2001. ACM Press.

[35] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, 1960.

[36] Mark Montague and Javed A. Aslam. Relevance score normalization for metasearch. In *CIKM '01: Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 427–433, New York, NY, USA, 2001. ACM Press.

[37] Mark Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 538–548, New York, NY, USA, 2002. ACM Press.

[38] C. N. Mooers. Information retrieval viewed as temporal signaling. In *Proceedings of the International Conference of Mathematicians*, pages 572–573, Cambridge, Massachusetts, 1952. American Mathematical Society.

[39] Martin F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

[40] Allison L. Powell, James C. French, Jamie Callan, Margaret Connell, and Charles L. Viles. The impact of database selection on distributed searching. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–239, New York, NY, USA, 2000. ACM Press.

[41] Edie Rasmussen. Evaluation in information retrieval. In *The MIR/MDL Evaluation Project White Paper Collection, Edition No. 3*, 2003.

[42] Yves Rasolofo, Faï¿$\frac{1}{2}$a Abbaci, and Jacques Savoy. Approaches to collection selection and results merging for distributed information retrieval. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 191–198, New York, NY, USA, 2001. ACM Press.

[43] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.

[44] G. Salton. *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall, Eaglewood Cliffs, NJ, USA, 1971.

[45] Gerard Salton. The smart document retrieval project. In *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 356–358, New York, NY, USA, 1991. ACM Press.

[46] Gerard Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036, 1983.

[47] Gerard Salton and Michael E. Lesk. Computer evaluation of indexing and text processing. *J. ACM*, 15(1):8–36, 1968.

[48] Tefko Saracevic and Paul Kantor. A study of information seeking and retrieving. iii. searchers, searches, and overlap. *Journal of the American Society for Information Science*, 39(3):197–216, 1988.

[49] E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, (January–February):11–14, 1997.

[50] Luo Si and Jamie Callan. Using sampled data and regression to merge search engine results. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, New York, NY, USA, 2002. ACM Press.

[51] Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz. Analysis of a Very Large AltaVista Query Log. Technical Report 1998-014, Digital SRC, 1998. http://gatekeeper.dec.com/pub/DEC/SRC/technical-notes/abstracts/src-tn-1998-014.html.

[52] C. W. Simpson and L. Prusak. Troubles with information overload–moving from quantity to quality in information provision. *International Journal of Information Management*, 15(6):413–425, December 1995.

[53] Charles L. Viles and James C. French. Dissemination of collection wide information in a distributed information retrieval system. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 12–20, New York, NY, USA, 1995. ACM Press.

[54] Charles L. Viles and James C. French. On the update of term weights in dynamic information retrieval systems. In *CIKM '95: Proceedings of the fourth international conference on Information and knowledge management*, pages 167–174, New York, NY, USA, 1995. ACM Press.

[55] Christopher C. Vogt and Garrison W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, 1999.

[56] Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning collection fusion strategies. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 172–179, New York, NY, USA, 1995. ACM Press.

[57] Ellen M. Voorhees, Narendra Kumar Gupta, and Ben Johnson-Laird. The collection fusion problem. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 95–104, 1994.

[58] Ellen M. Voorhees and Donna Harman. Overview of the fifth Text REtrieval Conference (TREC-5). In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 1–28, 1996.

[59] Ellen M. Voorhees and Donna Harman. Overview of the sixth Text REtrieval Conference (TREC-6). *Inf. Process. Manage.*, 36(1):3–35, 2000.

[60] Ellen M. Voorhees and Richard M. Tong. Multiple search engines in database merging. In *Proceedings of the Second ACM International Conference on Digital Libraries*, pages 93–102, Philadelphia, Pa., 1997. ACM Press, New York.

[61] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, December 1945.

[62] Ross Wilkinson and Philip Hingston. Using the cosine measure in a neural network for document retrieval. In *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 202–210, New York, NY, USA, 1991. ACM Press.

[63] Shengli Wu and Fabio Crestani. Data fusion with estimated weights. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 648–651, New York, NY, USA, 2002. ACM Press.

[64] Shengli Wu and Fabio Crestani. Shadow document methods of results merging. In *SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 1067–1072, New York, NY, USA, 2004. ACM Press.

[65] Jinxi Xu and Jamie Callan. Effective retrieval with distributed collections. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–120, 1998.

[66] Jinxi Xu and W. Bruce Croft. Cluster-based language models for distributed retrieval. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 254–261, New York, NY, USA, 1999. ACM Press.

[67] L.A. Zadeh. Fuzzy sets. In *Dubois, Prade, Yager (Eds.), Readings in Fuzzy Sets for Intelligent Systems*, pages 27–65, Los Altos, CA, USA, 1993. Morgan Kaufmann Publishers, Inc.