ProbFuse: A Probabilistic Approach to Data Fusion

David Lillis
School of Computer Science
and Informatics
University College Dublin
david.lillis@ucd.ie

Fergus Toolan
School of Computer Science
and Informatics
University College Dublin
fergus.toolan@ucd.ie

Rem Collier
School of Computer Science
and Informatics
University College Dublin
rem.collier@ucd.ie

John Dunnion
School of Computer Science
and Informatics
University College Dublin
john.dunnion@ucd.ie

ABSTRACT

Data fusion is the combination of the results of independent searches on a document collection into one single output result set. It has been shown in the past that this can greatly improve retrieval effectiveness over that of the individual results.

This paper presents *probFuse*, a probabilistic approach to data fusion. *ProbFuse* assumes that the performance of the individual input systems on a number of training queries is indicative of their future performance. The fused result set is based on probabilities of relevance calculated during this training process. Retrieval experiments using data from the TREC ad hoc collection demonstrate that *probFuse* achieves results superior to that of the popular CombMNZ fusion algorithm.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

information retrieval, data fusion, *probFuse*

1. INTRODUCTION

In the past, many algorithms have been developed to address the Information Retrieval (IR) task of identifying which documents in a database are most relevant to a given topic or query. More recently, researchers have focussed on attempting to improve upon the performance of individual IR models by combining the outputs of a number of such models into a single result set [15] [19] [24].

The task of fusing result sets produced by using a number of IR models to query the same document collection has become known as *data fusion* [1]. This is different to *collection fusion* [24], which involves the fusion of result sets that have been produced from querying distinct document collections that have little or no overlap.

This paper is organised as follows: In section 2 we describe the problem that fusion is intended to solve. Section 3 outlines previous work that has been undertaken by other researchers in this field. In section 4 we describe *probFuse*, a novel probabilistic algorithm for data fusion. Section 5 describes our experiments to evaluate the performance of the *probFuse* algorithm on inputs taken from two Text REtrieval Conferences (TREC). We also compare this with the performance of the popular CombMNZ approach [9]. Finally, our conclusions and future work are outlined in section 6.

2. PROBLEM DESCRIPTION

Of the numerous approaches to IR that have been proposed, none has been shown to achieve superior performance to all others in all situations. This may be as a result of difference in policies regarding query or document preprocessing, the algorithms used and representations of documents and queries. Individual IR systems have been shown to retrieve different documents in response to the same queries when operating on the same document collection [8]. This has been observed even where the overall retrieval effectiveness of these different systems has been similar [11].

Retrieval performance has been shown to be improved by fusing the result sets produced by a number of different IR systems into a single result set. A number of different approaches to data fusion are outlined in section 3.

Vogt and Cottrell [23] identify three "effects", any of which can be leveraged by a fusion technique. In some cases, a number of input result sets agree on the relevance of particular documents. Fusion techniques that take this agreement into account when compiling the fused result set will perform well in such circumstances. This is described as the "Chorus Effect". Experiments carried out by Lee [15] have shown that this is a very significant effect for data fusion tasks.

The exploitation of the Chorus Effect is the principal difference between the data fusion and collection fusion tasks. For data fusion, each individual IR technique is searching an identical document collection. This means that whenever a document is contained in multiple result sets, this can be presumed to infer relevance. However, when the document collections being searched are disjoint (collection fusion), this situation clearly cannot arise. In situations where the collections are partially overlapping, the presence of a document in multiple result sets cannot be used as an indication of greater relevance than a document that only appears in one. A document may only appear in a single result set because either the other IR models did not consider it to be relevant to the given query or it was not contained in the other document collections. Thus, in order for a fusion technique to make use of the Chorus Effect, it must be known that the document collections that are being queried by the different inputs have a very high degree of overlap.

They also describe the "Skimming Effect". Multiple result sets are more likely to result in higher recall (i.e. the fraction of relevant documents that have been retrieved) than a single one. A fusion technique can take advantage of this by "skimming" the top documents from each result set, as that is where the relevant documents are most likely to occur.

The "Dark Horse Effect" is an apparent contradiction to the Chorus Effect. Here, fusion effectiveness can be improved by identifying one input result set whose quality is of a substantially different level to the others. This may be due to either unusually high or unusually low numbers of relevant documents being returned. The apparent contradiction arises because while the Chorus Effect argues that fusion techniques should take as many of the input result sets as possible into account, the Dark Horse Effect argues in favour of identifying a single input result set

3. BACKGROUND RESEARCH

There are two principal categories of fusion techniques. Some algorithms make use of the score assigned to each document in each input result set to calculate a final score for that document in the fused result set. Because these raw scores are not always directly comparable (e.g. one input result set might assign scores in a range of 0-100 while another uses 0-1), score-based techniques frequently make use of a score normalisation phase before fusion takes place. This typically involves the mapping of all scores to a common range. Others make use of the rank each document occupies in each of the inputs, as the scores are not always available.

A Linear Combination model has been used in a number of studies [3] [6] [20] [23]. Under this model, a weight is calculated for each input model. In order to calculate a ranking score for a document, the score it is assigned by each input model is multiplied by that model's weight. These are then summed to get the final ranking score for the fused result set. A variation on the Linear Combination model using normalised scores was used in [13] and [22].

A number of fusion techniques based on normalised scores were proposed by Fox and Shaw [9]. Of these, CombSum and CombMNZ were shown to achieve the best performance and have been used in subsequent research. Under CombSum, a document's score is calculated by adding the normalised scores returned by the individual input models. Its CombMNZ score is found by multiplying the CombSum score by the number of non-zero relevance scores that it was assigned. In particular, Lee [15] achieved positive results using CombMNZ on the TREC-3 data set. These techniques have also been used in real-world systems: The *MetaCrawler* [21] and *SavvySearch* [12] meta search engines both use CombSum to fuse results.

Research by Manmatha et al. [17] demonstrated that the scores given to documents by an IR system can be modelled using a normal distribution for relevant documents and an exponential distribution for nonrelevant documents. This was possible even when relevance judgments were not available for the queries in question. Using Bayes' Rule, it was then possible to calculate the probability of relevance, given the score. When performing fusion, these probabilities were averaged, producing performance approaching that of CombMNZ.

Perhaps the simplest rank-based fusion technique is *interleaving* [24]. Under this system, the fused result set is constructed by firstly taking the top-ranked document from each input result set, followed by the second-ranked documents and so on. This approach operates on the assumption that each of the inputs are of similar effectiveness, and has been shown empirically to fall short of its goal of outperforming its inputs [25]. Voorhees et al. [25] proposed two variations on simple interleaving, in which training data was used to weight the input models according to past performance. At the interleaving stage, different quantities of documents were taken from each result set, depending on these weights, rather than taking equal amounts from each.

Lee [15] proposed a rank-based variation on CombMNZ, in which a function of each document's rank in each input result set was used as an alternative for normalised scores.

Aslam and Montague compared the fusion process to a democratic election in which there are few voters (the input models) and many candidates. They achieved positive results by implementing adapted implementations of two algorithms designed for that situation. *Borda-fuse* [2] awards a score to each document depending on its position in each input result set, with its final score being the sum of these. *Condorcet-fuse* [18] ranks documents based on a pairwise comparison of each. A document is ranked above another if it appears above it in more input result sets.

Other techniques have been proposed that make use of the actual textual content of the documents returned [7] [14]. Others rely on the individual input models providing metadata about the returned documents, other than simply a ranked list with relevance scores [10].

4. PROBABILISTIC FUSION

In this section, we describe *probFuse*, a probabilistic approach to data fusion. *ProbFuse* ranks documents based on their probability of relevance to the given query. This probability is calculated during a training phase, and depends on which input system returned the document amongst its results and the position in the result set in which the document was returned.

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. In order to run *probFuse*, we first build a set of probabilities for each input set. These probabilities are calculated by analysing the performance of each individual model on a number of training queries.

Rather than using the exact position a document occupies in each result set, the input result sets are divided into x segments. For each segment, the probability that a document being returned in this segment is relevant to a given query is calculated. This probability is averaged over $t\%$ of the total queries that are available.

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:

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

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.

In the past, it has been demonstrated that *probFuse* achieves significantly better results than CombMNZ when applied to small document collections [16]. For these collections, full relevance judgments are available, so the relevance of every document is known during the training phase. For larger collections, however, this is not the case, as the relevance judgments are incomplete (i.e. for some documents, it is unknown whether they are relevant or nonrelevant to the given queries). For this reason, we also use a slight variation of the probability calculation. This allows us to observe the effects, if any, of different methods of dealing with unjudged documents.

Equation 1 takes all the documents in a segment into account, assuming unjudged documents to be nonrelevant. Our modified probability calculation ignores unjudged documents and thus only takes into account documents that have been judged to be either relevant or nonrelevant. In this case, the probability $P(d_k|m)$ is given by

$$P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{|R_{k,q}|}{|R_{k,q}|+|N_{k,q}|}}{Q} \tag{2}$$

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.

We refer to *probFuse* runs using the probability calculation in equation 1 as *probFuseAll* and those using equation 2 as *probFuseJudged*. From these equations, it can be seen that for document collections with complete relevance judgments, the probabilities calculated by *probFuseAll* and *probFuseJudged* will be identical.

After the training phase is complete and a set of probabilities for each input model has been built, we can then use this to construct a fused result set for subsequent queries. For these, the ranking score S_d for each document d is given by

$$S_d = \sum_{m=1}^{M} \frac{P(d_k|m)}{k} \tag{3}$$

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 model 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 return them as being judged relevant.

This approach to data fusion attempts to make use of the three effects described in section 2 above. By using the sum of the probabilities, we attach more significance to documents that have been returned by multiple input models, 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 model, we attach greater importance to input models that are more likely to return relevant documents in particular segments (Dark Horse Effect).

5. EXPERIMENTS AND EVALUATION

In this section, we describe the experiments that were performed to evaluate the effectiveness of *probFuse*. The *probFuse* algorithm was applied to a number of different combinations of input result sets and the resulting fused result was compared to that of the popular CombMNZ algorithm. CombMNZ is easily implemented and has been shown to perform well on data fusion tasks [15]. This has made it an attractive choice when choosing a baseline technique to compare with. As such, it has become the standard algorithm to use [4] [19].

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. A number of different normalisation strategies have been proposed. We have chosen the one used by Lee [15], as it is the one most commonly used for comparison and has been described as "Standard Normalisation" [18]. Lee's implementation of CombMNZ calculates normalised scores using

$$normalised\_sim = \frac{unnormalised\_sim - min\_sim}{max\_sim - min\_sim} \tag{4}$$

where max_sim and min_sim are the maximum and minimum scores that are actually seen in the input result set. Once the scores have been normalised, $CombMNZ_d$, the CombMNZ ranking score for any document d is given by

$$CombMNZ_d = \sum_{s=1}^{S} N_{s,d} \times |N_d > 0| \tag{5}$$

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.

5.1 Experimental Setup

As our inputs, we used data from the ad hoc retrieval track of the TREC-3 and TREC-5 conferences. This data consists of the topfile (a collection of result sets) produced by each of the groups that participated in those conferences. Each topfile contains result sets for 50 topics (queries): TREC-3 uses TREC topics 151-200 and TREC-5 uses topics 251-300. 40 topfiles are available for TREC-3, while 31 are available

Table 1: Inputs to TREC-3 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 2: Inputs to TREC-5 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

for TREC-5. Only the topfiles in Category A were considered for TREC-5, as Category B participants operated on only a subset of the data.

For each of these two data sets, we performed 5 experimental runs. Each experimental run firstly involved choosing six random topfiles to use as inputs. In order to eliminate the results being skewed by the ordering of the topics, we produced five random orderings for the topics and performed data fusion using both *probFuse* and CombMNZ over each of these. The performance evaluation values for each run are the average for each of those five random orderings. No input topfile was used in multiple runs. The inputs used for each of the five runs for TREC-3 and TREC-5 are shown in Table 1 and Table 2 respectively. The inclusion of this list of inputs is intended to aid the reproduction of our experiments.

Each run was performed for a variety of training set sizes defined as a percentage of the number of available queries. In the case of the TREC-3 and TREC-5 data, the number of available queries is always 50. The number of segments into which each result set was divided was also varied. We used training set sizes, t such that $t \in \{10, 20, 30, 40, 50\}$ and numbers of segments, x such that $x \in \{2, 4, 6, 8, 10, 15, 20, 25, 30, 40, 50, 100, 150, 200, 250, 300, 400, 500\}$.

In order to ensure comparability, CombMNZ was only applied to those queries used in the fusion phase of *probFuse*, with the training queries being ignored. This will cause the evaluation results for CombMNZ to vary as the training set size changes, since the number of remaining queries on which fusion is performed also changes.

The goal of these experiments is to empirically determine the combination of training set size and number of segments (denoted by x) that achieves the greatest retrieval performance, both in terms of the evaluation scores it receives and its performance in relation to CombMNZ. We approach this by firstly identifying a training set size that results in high performance for both the TREC-3 and TREC-5 input sets. This is discussed in section 5.2. Once this is done, we examine the performance of both variations of *probFuse* for different values of x to find an x-value that performs well on both input sets when averaged over the five runs. This is done in section 5.3.

The evaluation of the fused output result sets was performed by *trec_eval*, which is the evaluation software used for the TREC conferences [26]. We use two evaluation measures in our experiments. Firstly, we use Mean Average Precision (MAP) to find our training set size and x-value. MAP is 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 [5]. Documents for which a relevance judgment is not available are considered to be nonrelevant.

After identifying this training set size and x-value, we then examine each of the five experimental runs to make a comparison with the CombMNZ algorithm in a more detailed manner. At this stage, in addition to MAP we also make use of the bpref evaluation measure. Bpref only takes judged documents into account and is inversely related to the fraction of judged nonrelevant documents that are retrieved before relevant documents [5]. The analysis of these evaluation results is contained in section 5.4.

5.2 Training Set Size

Initially, the MAP measure was used to identify which training set sizes resulted in the best performance. Table 3 and Table 4 show the average MAP achieved when using each training set size, along with its improvement over the corresponding figure for CombMNZ. The MAP score included in those figures is the average MAP score for all values of x (the number of segments) at that training set size over all five runs. The average MAP scores for CombMNZ vary with training set size, despite CombMNZ not making use of any training phase. In each of those tables, the highest MAP score and the greatest improvement over CombMNZ for each of the *probFuse* variants are marked in bold type.

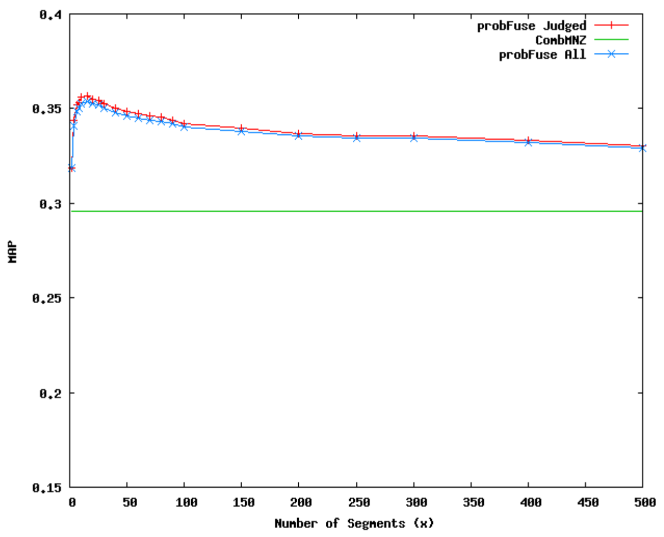

Figure 1: TREC-3 MAP scores for $t = 50\%$ (average over 5 runs)

From Table 3, we can see that the highest average MAP score by both *probFuse* variations on the TREC-3 inputs was achieved using a training set size of 30%. The biggest per-

Table 3: 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%**

Table 4: 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%

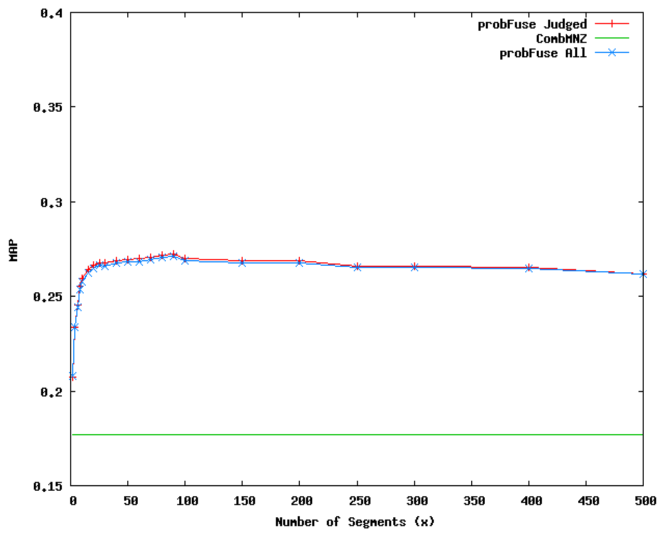

Figure 2: TREC-5 MAP scores for $t = 50\%$ (average over 5 runs)

centage increase over CombMNZ was when using a training set size of 50%. For the TREC-5 inputs (seen in Table 4), we note that the highest average MAP score was when using a training set size of 50%. The greatest percentage improvement over CombMNZ was achieved for training set sizes of 20% and 30% for *probFuseAll* and *probFuseJudged* respectively.

Both variations of *probFuse* achieve higher average MAP scores than CombMNZ for every training set size. This is the case for both the TREC-3 and TREC-5 input sets.

For the purposes of Section 5.3, we have chosen to use a training set size of 50%. At that level, *probFuseAll* and *probFuseJudged* both achieve either their highest MAP score or their highest increase over CombMNZ for both sets of inputs. This would not be the case had we chosen a training set size of 30%, as *probFuseAll* achieves its biggest improvement over CombMNZ at a training set size of 20%. At 50%, both *probFuse* variations achieve their highest average MAP scores on the TREC-5 inputs. For the TREC-3 inputs, both variations achieve their highest improvement over CombMNZ, and their average MAP scores are within 2% of the highest they achieve at any level.

5.3 Number of Segments

Figure 1 and Figure 2 show the MAP scores for each number of segments for a training set size of 50% for the TREC-3 and TREC-5 inputs respectively. Each of these MAP scores is the average of the MAP scores achieved in each of the five runs.

It is interesting to note that *probFuseJudged* and *probFuseAll* both show near-identical results for both input sets. In each graph, performance is at its worst for a value of $x = 2$. This is to be expected, as in that situation, each result set is being divided into only two segments, so the probability of relevance being assigned to each document is based on whether it appears in the first or second half of the result set, which is too coarse a measure. Initially, as x increases, the average MAP score improves. A gradual decline is then seen for higher values of x.

The principal difference in the trend in the two graphs is in the point at which the average MAP score reaches its peak. This peak is reached for a much lower x-value on the TREC-3 inputs, where $x = 15$. For TREC-5, the MAP score continues improving gradually until the point where $x = 90$. Thereafter, both show a downward trend as x increases.

An x-value of 25 yields the best MAP score, on average, over both input sets. For that reason, this is the x-value we have chosen to use when analysing the individual runs in section 5.4.

5.4 Analysis of Individual Runs

Having identified a high-performing training set size (50%) and number of segments to divide each result set into (25), we can examine the five individual experimental runs in more detail.

Table 5 shows the results of the five individual runs on the TREC-3 input set. In that table, figures in parenthe-

Table 5: **TREC-3 performance of five individual runs for $t = 50\%$ and $x = 25$**

	CombMNZ		*probFuseAll*		*probFuseJudged*	
	MAP	bpref	MAP	bpref	MAP	bpref
first	0.16726	0.23960	0.30988 (+85.27%)	0.31458 (+31.29%)	0.31144 (+86.20%)	0.31628 (+32.00%)
second	0.28752	0.33434	0.34100 (+18.60%)	0.33118 (-0.95%)	0.34402 (+19.65%)	0.33356 (-0.23%)
third	0.43344	0.41222	0.41348 (-4.61%)	0.39222 (-4.85%)	0.41620 (-3.98%)	0.39416 (-4.38%)
fourth	0.23416	0.31048	0.30374 (+29.71%)	0.30314 (-2.36%)	0.30766 (+31.39%)	0.30528 (-1.67%)
fifth	0.35548	0.39616	0.39108 (+10.01%)	0.38006 (-4.06%)	0.39294 (+10.54%)	0.38308 (-3.30%)
Average	0.29557	0.31707	0.35184 (+19.04%)	0.34804 (+9.77%)	0.35445 (+19.92%)	0.35046 (+10.53%)

ses represent the percentage difference to the corresponding score for CombMNZ. Figure 3 shows the MAP scores for CombMNZ, *probFuseAll* and *probFuseJudged* for each run on the TREC-3 data, and figure 4 shows the bpref scores for TREC-3.

Both variants of *probFuse* achieved a higher MAP score than CombMNZ for all runs except for "third". On that run, CombMNZ scored slightly higher. It is important to highlight that both *probFuse* variations actually achieve their highest MAP scores for that run. The MAP score for CombMNZ is also the highest it achieves for any run. The lower MAP score achieved by *probFuse* on the "third" run can therefore be attributed to an unusually high MAP score being achieved by CombMNZ on that run, rather than *probFuse* underperforming.

Using the bpref measure, the performance of *probFuse* is slightly below that of CombMNZ for four of the five runs. For this reason, it cannot be said that *probFuse* outperforms CombMNZ on the TREC-3 inputs under bpref. However, neither *probFuse* variant drops below 95% of CombMNZ's bpref score on any run, and for the "first" run, *probFuse* achieves a vastly superior result, leading to a better average performance.

Under both the MAP and bpref measures, *probFuseJudged* achieves higher performance than *probFuseAll* on each of the five experimental runs. However, this increase only exceeds 1% for the MAP score on the "fourth" run, and never exceeds 2%.

Table 6 shows a similar table detailing the five individual experimental runs on the TREC-5 input set. Figure 5 shows the MAP scores for CombMNZ, *probFuseAll* and *probFuseJudged* for each run on the TREC-5 data, and figure 6 shows the bpref scores for each fusion technique.

Here, *probFuse* outperforms CombMNZ on each of the runs using both evaluation measures. In particular, runs "second", "third" and "fifth" show large performance gains for both *probFuse* variations over CombMNZ for both the MAP and bpref measures.

As with TREC-3, *probFuseJudged* performs better than *probFuseAll*, although once again the degree of improvement is less than 1% in almost all cases. The exception to this is the MAP score for the "first" run, which is the only case where *probFuseAll* performs better than *probFuseJudged*. This is particularly interesting in the case of the bpref scores, as bpref only takes judged documents into account, ignoring documents for which relevance judgments are not available. For this level of incompleteness, the performance of *probFuseAll* is similar to that of *probFuseJudged*. It is left to future work to determine if this remains the case as the available relevance judgments become more incomplete.

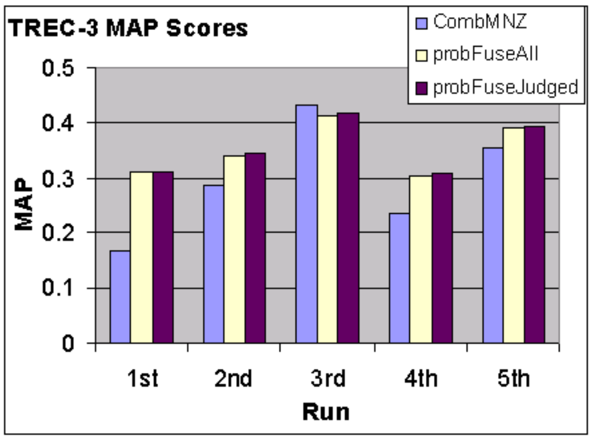

Figure 3: TREC-3 MAP scores for $t = 50\%$ and $x = 25$

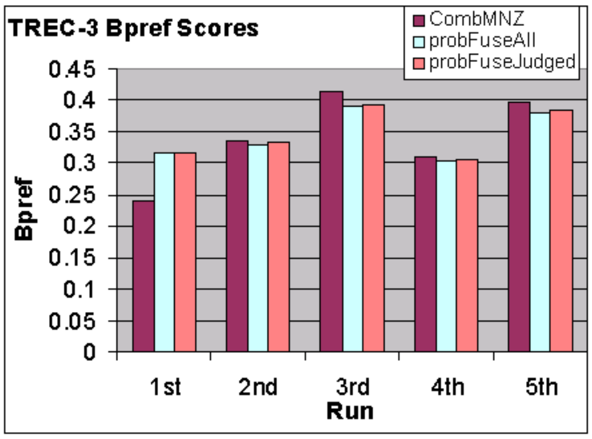

Figure 4: TREC-3 bpref scores for $t = 50\%$ and $x = 25$

The performance on the TREC-3 inputs is superior to CombMNZ when evaluated using the MAP measure, with the bpref scores falling only slightly below those of CombMNZ in some cases. For the TREC-5 inputs, *probFuse* has shown significant improvement over CombMNZ when evaluated by both MAP and bpref.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have described *probFuse*, a data fusion algorithm that relies on the probability of relevance to cal-

Table 6: TREC-5 performance of five individual runs for $t = 50\%$ and $x = 25$

	CombMNZ		*probFuseAll*		*probFuseJudged*	
	MAP	bpref	MAP	bpref	MAP	bpref
first	0.25144	0.26406	0.27378 (+8.88%)	0.26814 (+1.55%)	0.26264 (+4.45%)	0.26878 (+1.79%)
second	0.22480	0.26896	0.35560 (+58.19%)	0.33918 (+26.11%)	0.35844 (+59.45%)	0.34140 (+26.93%)
third	0.12306	0.19232	0.26838 (+118.09%)	0.24744 (+28.66%)	0.27050 (+119.81%)	0.24920 (+29.58%)
fourth	0.12626	0.14520	0.15546 (+23.13%)	0.15734 (+8.36%)	0.15602 (+23.57%)	0.15746 (+8.44%)
fifth	0.16004	0.21790	0.27842 (+73.97%)	0.26474 (+21.50%)	0.27922 (+74.47%)	0.26498 (+21.61%)
Average	0.17712	0.19740	0.26633 (+50.37%)	0.25537 (+29.37%)	0.26787 (+51.24%)	0.26212 (+32.79%)

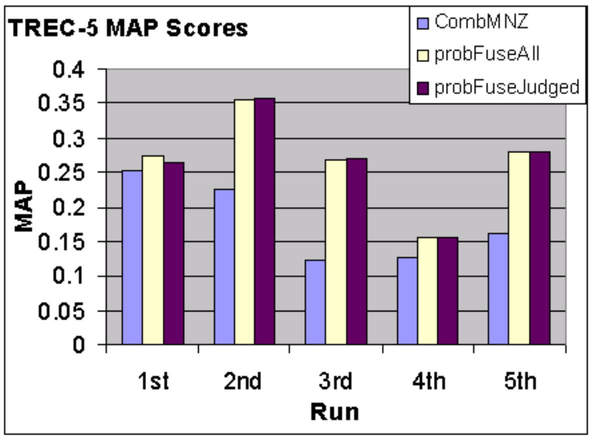

Figure 5: TREC-5 MAP scores for $t = 50\%$ and $x = 25$

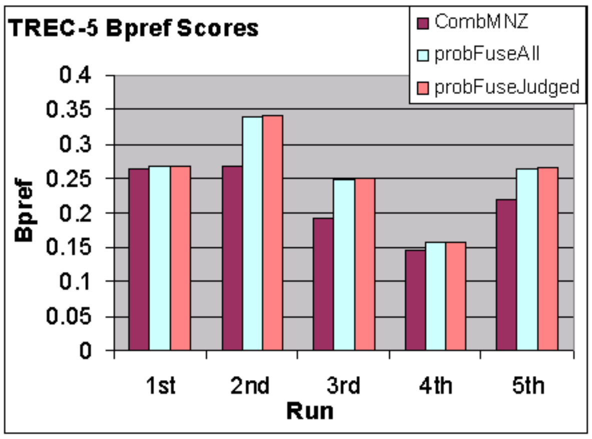

Figure 6: TREC-5 bpref scores for $t = 50\%$ and $x = 25$

culate a ranking score for documents in a fused result set. These probabilities are calculated based on the position of relevant documents in result sets returned in response to a number of training queries.

In experiments using data from the ad hoc track of the TREC-3 and TREC-5 conferences, two variations of *probFuse* were shown to significantly outperform the popular CombMNZ algorithm over a number of different combinations of inputs. *ProbFuseAll* achieved a MAP score that was, on average, 19% higher than CombMNZ on the TREC-3 inputs and 50% higher on TREC-5. Similarly, the average bpref score was 10% higher than CombMNZ on TREC-3 inputs and 29% higher on TREC-5.

These results follow on from experiments on small document collections for which complete relevance judgments are available [16]. Due to the incomplete nature of the relevance judgments for TREC-3 and TREC-5, we also tested a variation of *probFuse*, called *probFuseJudged*, that only takes judged documents into account when calculating its probabilities. *ProbFuseJudged* achieved an increase of 11% over CombMNZ on the TREC-3 inputs and an increase of 31% on TREC-5. The MAP scores it achieved were 20% higher than CombMNZ on TREC-3 and 51% higher on the TREC-5 data.

It is interesting to note that *probFuseJudged* only achieved marginal performance gains over *probFuseAll*, even using the bpref evaluation measure, which only takes judged documents into account.

Our future work will apply *probFuse* to larger collections with greater levels of incompleteness in their available relevance judgments (e.g. the Web track collections of later TREC conferences). We intend to investigate whether the performance of *probFuseJudged* and *probFuseAll* diverges as the level of incompleteness increases. In addition, the relevance judgments for some collections differentiate between different degrees of relevance (e.g. for the WT10G collection, documents can be judged nonrelevant, relevant and highly relevant). We also intend to investigate whether adjustments to our probability calculations that take this information into account will be beneficial.

Another potential research direction is to investigate the possibilities of applying *probFuse* to a document collection without the necessity of using training data from that collection. This could potentially involve the use of the techniques outlined by Manmatha et al. [17] to estimate the probability of relevance, or alternatively training *probFuse* on one document collection in order to apply it to another.

7. REFERENCES

[1] J. A. Aslam and M. 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] J. A. Aslam and M. 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] B. T. Bartell, G. W. Cottrell, and R. 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.

[4] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, O. Frieder, and N. Goharian. Fusion of effective retrieval strategies in the same information retrieval system. *J. Am. Soc. Inf. Sci. Technol.*, 55(10):859–868, 2004.

[5] C. Buckley and E. 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.

[6] J. P. Callan, Z. Lu, and W. B. 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.

[7] N. Craswell, D. Hawking, and P. B. Thistlewaite. Merging results from isolated search engines. In *Australasian Database Conference*, pages 189–200, Auckland, New Zealand, 1999.

[8] P. Das-Gupta and J. 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.

[9] E. A. Fox and J. 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.

[10] L. Gravano, K. Chang, H. Garcia-Molina, and A. Paepcke. Starts: Stanford protocol proposal for internet retrieval and search. Technical report, Stanford, CA, USA, 1997.

[11] D. 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.

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

[13] L. S. Larkey, M. E. Connell, and J. 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.

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

[15] J. H. Lee. Analyses of multiple evidence combination. *SIGIR Forum*, 31(SI):267–276, 1997.

[16] D. Lillis, F. Toolan, A. Mur, L. Peng, R. Collier, and J. 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, Portstewart, Northern Ireland, 2005. University of Ulster.

[17] 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.

[18] M. Montague and J. 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.

[19] M. Montague and J. 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.

[20] A. L. Powell, J. C. French, J. Callan, M. Connell, and C. 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.

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

[22] L. Si and J. 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.

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

[24] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. The collection fusion problem. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 95–104, 1994.

[25] E. M. Voorhees, N. K. Gupta, and B. 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.

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