Improving High Accuracy Retrieval by Eliminating the Uneven Correlation Effect in Data Fusion

Shengli Wu and Sally McClean
University of Ulster, UK
E-mail: {s.wu1, si.mcclean}@ulster.ac.uk

November 2, 2005

Abstract

The aim of this research is twofold. On the one hand, high accuracy retrieval has been a concern of the information retrieval community for some time. We aim to investigate this issue via data fusion. On the other hand, correlation among component results has been proven to be harmful to data fusion, but has not been taken into account in data fusion algorithms. In the hope of achieving better performance, we propose a group of algorithms to eliminate the effect of uneven correlation among component results by assigning different weights to all component results or their combinations. Then the linear combination method or a variation is used for fusion.

Extensive experimentation is carried out to evaluate the performances of these algorithms with six groups of results, which are the top ten systems submitted to TREC 6, 7, 8, 9, 2001, and 2002. The experimental results show that all eight data fusion methods involved outperform the best component system on average. Therefore, it demonstrates that the data fusion technique in general is effective with accurate retrieval results. The experimental results also demonstrate that all six methods presented in this paper are effective on eliminating the effect of uneven correlation among component results. All of them outperform CombSum and five of them outperform CombMNZ on average.

Keywords: information retrieval, data fusion, results correlation, experimentation, performance

1 Introduction

In the last couple of years the data fusion issue has been investigated by many researchers in the information retrieval field. The starting point is: for the same information need, different information retrieval systems retrieve different sets of documents (usually some overlap does exist) from the same document
collection. The difference between these information retrieval systems could be diversified: different query representations, different document representations, different retrieval strategies; difference may also include factors such as parsing rules, stemming, phrase processing, relevance feedback techniques, etc. Researchers try to merge these results from multiple systems for better retrieval effectiveness. This provides an alternative method for implementing an effective information retrieval system by taking advantage of data fusion technique (e.g., in (Shaw & Fox, 1995; Ng, Loewenstern, Basu, Hirsh, & Kantor, 1996)). In addition, meta-search engines in the context of WWW appear to be a relevant application for data fusion, and some results for data fusion may be useful here as well.

Usually, the performance of information retrieval (ad-hoc retrieval) is measured with precision and recall or some variations of them. In previous research, measures such as mean average precision, average precision at certain cut-off document levels, and R-precision have been used for the evaluation of data fusion algorithms. Alternatively, to achieve high accuracy is the most important aspect for question answering. Recently, recognizing the importance of achieving high accuracy in some applications such as mobile web browsing and information analysis, TREC introduced a new track called High Accuracy Retrieval from Documents (HARD) in 2003 (Allan, 2001). Some researchers also have considered the possibility of achieving high accuracy in ad-hoc retrieval. For example, Shah and Croft (Shah & Croft, 2004) proposed several such methods by incorporating approaches from question answering. In such a situation, the focus is to try to get the first relevant result as high as possible in the ranked list. The Mean Reciprocal Rank (MRR), instead of recall and precision, is a suitable measure for evaluating the performance of high accuracy retrieval.

In this paper, we aim to investigate the issue of high accuracy retrieval via data fusion. Previous research on data fusion has demonstrated the effectiveness of a number of data fusion algorithms such as CombSum, CombMNZ, the linear combination method and others (Fox, Koushik, Shaw, Modlin, & Rao, 1993; Fox & Shaw, 1994; Lee, 1997; Ng & Kantor, 2000; Vogt & Cottrell, 1998, 1999). In many cases, the fused results using these methods are better than the best component retrieval results. Now we would like to see how these data fusion methods perform for high accuracy retrieval. We hope to provide a better understanding of such issues through extensive experimentation.

Besides some existent data fusion algorithms such as CombSum, CombMNZ, and linear combination, we would like to investigate data fusion algorithms which consider correlation between component retrieval results. If some of the retrieval results involved in data fusion correlate more strongly than the others, their common opinion will dominate the voting process in data fusion. This may degrade the effectiveness of data fusion in many cases, especially when very good results appear to be a minority. Results correlation has been identified as one of the major aspects which affect data fusion significantly (Vogt & Cottrell, 1998, 1999; Ng & Kantor, 2000; Wu & McClean, ). However, how to eliminate the harmful effect of the uneven correlation among results (presumably we have at least three component results for data fusion) has not been investigated.
In the light of the above issues, we study to which extent data fusion can help in getting high accuracy for ad-hoc retrieval. In particular, six data fusion algorithms are presented for eliminating the harmful effect of uneven correlation among component results. Extensive experimentation is carried out to evaluate their performances.

The rest of this paper is organized as follows: in Section 2 we review some related work. In Section 3 we describe six data fusion methods which aims to eliminate the uneven correlation effect among results. In Section 4 we present the experimental setting and results for the evaluation of the proposed methods. Section 5 discusses some more observations in the experiments. Section 6 concludes the paper.

2 Previous work on data fusion

There has been quite a large body of research on data fusion in the field of information retrieval. Some early related work on data fusion is from Turtle and Croft (Turtle & Croft, (1991)), Foltz and Dumais (Foltz & Dumais, 1992), and Belkin and his colleagues (Belkin, Cool, Croft, & Callan, 1993; Belkin, Kantor, Fox, & Shaw, 1995). Turtle and Croft (Turtle & Croft, (1991)) used independently-generated query representations to create a number of results within an inference network, and found that combining different query representations led to increased retrieval effectiveness over any single representation. Foltz and Dumais (Foltz & Dumais, 1992) found similar improvements by combining results from multiple retrieval strategies. Belkin and his colleagues (Belkin et al., 1993, 1995) conducted experiments with a 2GB TREC collection from TREC 1, and observed effectiveness improvement over a large number of combinations of different Boolean query representations. Later Saracevic and Kantor (Saracevic & Kantor, 1998) used independently-generated query representations to create a number of results, and found that a document was more likely to be relevant if it appeared in multiple results.

In the following let us review some more work addressing certain special issues.

2.1 Data fusion methods based on score information

Fox and his colleagues (Fox et al., 1993; Fox & Shaw, 1994) introduced a group of data fusion methods including CombSum and CombMNZ. CombSum sets the score of each document in the combination to the sum of the scores obtained by the component result, while in CombMNZ the score of each document is obtained by multiplying this sum by the number of results which have non-zero scores.

Bartell et al. investigated the linear combination method in (Bartell, Cottrrell, & Belew, 1994). Numerical optimization techniques were used to determine optimal weights for component systems and positive results were achieved.
Usually when using the linear combination method, some training queries and evaluations are required to determine the performance of those systems involved. This work demands a lot of human effort. For solving this problem, a few methods have been proposed to estimate the performance of a group of systems without any human judgment (Soboroff, Nicholas, & Cahan, 2001; Wu & Crestani, 2002; Amitay, Carmel, Lempel, & Soffer, 2004; Nuray & Can, ).

2.2 Score normalization

Usually, scores obtained from different information retrieval systems may be diverse and it is impossible to compare them directly. Therefore, some kinds of score normalization is required. In (Lee, 1997), Lee proposed the linear [0,1] normalization method among other things. In this method, for a list of documents in a result, the maximal score is transformed to 1, the minimal score is transformed to 0, and any other score is linearly transformed to a score between 0 and 1 accordingly.

Montague and Aslam (Montague & Aslam, 2001) suggested two other linear transformation methods Sum and ZMUV (Zero-Mean and Unit-Variance). In Sum, the minimal score is mapped to 0 and the sum of all scores in the result to 1. In ZMUV, the average of all scores is mapped to 0 and their variance to 1.

Manmatha and colleagues (Manmatha, Rath, & Feng, 2001) investigated the score distribution for a given query. They found that it might be modelled using an exponential distribution for the set of non-relevant documents and a normal distribution for the set of relevant documents. Also they investigated how to use this to improve the performance of data fusion.

2.3 Data fusion methods based on ranking information

Scores are not always available. For example, very few web search engines provide scores for the retrieved web documents. When only ranking information was available, Montague and Aslam suggested a few methods including Borda fusion (Aslam & Montague, 2003) and Condorcet fusion (Montague & Aslam, 2002).

In Borda fusion, for a set of $n$ documents in a list, the top-ranked document is given $n$ points, the second ranked document is given $n - 1$ points, and so on. Then CombSum is used for the merging process. Condorcet fusion was borrowed from political science for majority voting. It considers all possible head-to-head ranking competition among all possible document pairs. Then we can rank these documents according to the number of competitions they have won.

Using a Markov chain was investigated by some researchers (Dwork, Kumar, Naor, & Sivakumar, 2000; Renda & Straccia, 2003). Markov chain can be used to realize different data fusion algorithms, which depend on the specific Markov chain involved.

Wu and McClean (Wu & McClean, ) found that rank-probability of relevance relationship in resultant document list can fit very well with a logarithmic curve.
2.4 Performance prediction

Vogt and Cottrell (Vogt & Cottrell, 1998, 1999) analysed the performance of the linear combination algorithm by linear regression. In their experiments, they used all possible pairs of 61 systems submitted to TREC 5 ad-hoc track. The similarity of two results’ rankings and 13 other variables were used in the analysis. The performance analysis and prediction for the fused result was very accurate.

Ng and Kantor (Ng & Kantor, 2000) used several different statistical techniques to predict if the performance of CombSum is better than both component systems involved or not. Two variables were used: performance ratio of two systems, and a measure of the dissimilarity between two systems. They found that the two variables were informative to predict if the fused result was better than both component results.

Wu and McClean (Wu & McClean, 2000) analysed the performances of CombSum and CombMNZ with three groups of results submitted to TREC 6, TREC 2001 and TREC 2004. Multiple regression was used and four variables including the number of results, the overlap rate among the results, the mean average precision of the results, and the standard deviation of the mean average precision of the results were identified as highly significant metrics which affect the performance of data fusion. Based on the analysis, they predicted the performance of the data fusion methods such as CombSum and CombMNZ with variable number of results. The experiments showed that the prediction of the performance of data fusion was quite accurate.

2.5 Fusion mechanism analysis

Lee (Lee, 1997) did some initial work by conducting an experiment to support the hypothesis: different retrieval processes might retrieve similar sets of relevant documents but retrieve different sets of non-relevant documents. Furthermore, Lee stated that as long as the component results being used for fusion had greater relevant overlap than non-relevant overlap, improvement would be observed. That can explain why the multiple evidence fusion methods such as CombMNZ are very effective data fusion methods.

Beitzel et al. (Beitzel et al., 2004) conducted some experiments to compare the performances of CombMNZ using several different groups of systems. They observed no improvement when fusing results from three different retrieval strategies in the same information retrieval system, while the merged result was better than the best system when choosing the top three systems submitted to TREC 6, 7, 8, 9 and 2001.

Hsu and Taksa (Hsu & Taksa, 2005) compared rank and score data fusion methods via analysis and simulation. They found that the fusion using rank performs better than the fusion using score under certain conditions.
2.6 Web metasearch engines

Web meta-search engines can be regarded as a relevant application of data fusion techniques. However, it should be noted the document collections in every web search engine is not identical with each other, though heavy overlap is a possibility.

Many web meta-search engines take a different approach from those mentioned above to the fusion problem. Instead of relying on the engine’s ranking mechanism, Inquirus (Lawrence & Giles, 1998a, 1998b) retrieves the full contents of the web pages returned and ranks them more like a traditional search engine using information retrieval techniques applicable to full documents only. Similar to Inquirus, Mearf (Oztekin, Karypis, & Kumar, 2002) retrieved the full contents of the web pages returned, then re-ranked them based on the analytical results on contents and/or links of these web pages.

Beitzel et al. (Beitzel, Jensen, Frieder, Chowdhury, & Pass, 2005) suggested a surrogate scoring method for web meta-search engines. Instead of catching the full content of those documents, surrogates (titles, URLs, short snippets, and the rank positions) can be used to calculate scores for these documents, because score usually is not available from web search engines. Based on that, some regular data fusion methods such as CombSum and CombMNZ can be used for the merging.

The aim of our study is twofold: investigating high accuracy retrieval via data fusion and investigating the data fusion algorithms which can eliminate the harmful effect of uneven correlation among component results.

3 The algorithms of eliminating the uneven correlation effect in component results

From the experiments conducted by many researchers before (e.g., in (Beitzel et al., 2004; Smeaton, 1998; Vogt & Cottrell, 1998, 1999; Wu & McClean, ; ?)), we can conclude that the ideal situation for data fusion is:

- all component results are equal in performance;
- all component results have very weak correlation with each other;
- all pairs of component results have a equal strength of correlation.

In real applications, the above conditions may not be well satisfied at the same time by all component results involved; however, some remedies are possible. If component results are quite different in performance, then we can estimate their performance by training, assign a weight to each component result, and use the linear combination method for data fusion. If all component results are strongly correlated (it is very likely the case for several retrieval strategies in the same information retrieval system, as suggested by Beitzel et. al.’s experimental results (Beitzel et al., 2004)), then no effective remedy is available since
all the results are very much alike. The best thing we can do is to avoid this situation. If some of the component results are more strongly correlated then the others, then the opinion among the strongly correlated results will dominate the data fusion process. Such a phenomenon is potentially harmful to data fusion, since highly correlating results are very likely to over-promote some common non-relevant documents among them. To find effective data fusion methods to solve this problem is one of the aims of this study. The general idea is to assign correlation weights to all results (or/and) their combinations; then the linear combination method is used in the voting process.

In certain circumstances, divergent performance and uneven correlation among results may happen at the same time, and then a combined solution is desirable. It is straightforward to modify some of the algorithms presented in this paper for the situation of divergent performance.

3.1 Correlation Methods 1 and 2

First let us discuss how to calculate the strength of the correlation between two component results. In this study, we use the overlap rate of two results to indicate that, since our preliminary investigation in (Wu & McClean,) suggests that using either overlap rate or Spearman rank coefficient leads to similar outcome, and it is more efficient to calculate overlap rate than to calculate Spearman rank coefficient.

Suppose we have \( n \) \((n > 2)\) results \( r_1, r_2, ..., r_n \) for fusion, for every pair of results \( r_i \) and \( r_j \) \((1 \leq i \leq n, 1 \leq j \leq n, i \neq j)\), the overlap coefficient between two results is:

\[
v_{ij} = \frac{2 \cdot |o_{ij}|}{|r_i| + |r_j|}
\]

where \(|o_{ij}|\) is the number of overlapping documents in component results \( r_i \) and \( r_j \). \(|r_i|\) and \(|r_j|\) are the number of documents in result \( i \) and \( j \), respectively.

For every result \( r_i \), we obtain its average overlap coefficient with all other results \( w_i = 1 - \frac{1}{n-1} \sum_{j=1,2,...,n,j \neq i} v_{ij} \). In such a way, if a component result correlates more closely with all other results on average, then a lighter weight is assigned to it; otherwise, a heavier weight is assigned to it. Having \( w_i \) for every result \( r_i \), we use the linear combination method to achieve data fusion:

\[
s(w, d, q) = \sum_{i=1}^{n} w_i s_i(d, q)
\]

where \( s = (s_1(d, q), s_2(d, q), ..., s_n(d, q)) \) are the scores of document \( d \) in the fused result for a given query \( q \).

The above method is referred to as Correlation Method 1 later in this paper. If a result is very independent, then it is very likely to obtain a lower overlap coefficient than some other results, and this again leads to a heavier weight assigned to it and make the result more influential to data fusion; conversely, if a result is very dependent, then it is very likely to obtain a higher
overlap coefficient, and this again leads to a lighter weight assigned to it and makes it less influential on data fusion. Using this method, we do not need other information except for the component results as in CombSum and combMNZ.

In the above method, the weight of every result is calculated in every query, which may vary from one query to another randomly. A moderated method to this (we call it Correlation Method 2 later) is to calculate the weights of all results by using a group of training or historical query results and take the average values of them.

Suppose there are \( n \) results, and each result includes \( l \) documents, then calculating the overlap rate between any pair of results needs \( O(l) \) time if we use a hash table. There are \( n \times (n - 1)/2 \) possible pairs of results for a total of \( n \) results, and to calculate the overlap rates among all possible pairs takes \( O(l \times n^2) \) time, which is the time complexity of Correlation Methods 1 and 2 for weights assignment. With weights assigned, the complexity of data fusion process is the same as that of the linear combination method (\( O(l \times n) \)).

When some component results are much better than some other component results, it worths of consideration of their performance difference as well. It is straightforward to combine these two aspects. Suppose a result \( A \) is assigned a weight of \( w_p \) for its performance, and a weight of \( w_c \) for its correlation with other results, then a weight \( w = w_p \times w_c \) can be assigned to \( A \). All other results can be processed in the same way.

### 3.2 Correlation Methods 3 and 4

The general idea of Correlation Methods 3 and 4 is the same as that of Correlation Methods 1 and 2. The only difference is that we use a different method for weights assignment.

The weights assignment process is: first we merge all \( n \) results into a single set. For every merged document \( d \), a count number is used to indicate the number of results which includes \( d \). Furthermore, we also say that every corresponding document of \( d \) in \( r_1, r_2, \ldots, r_n \) has the same count number as \( d \) has.

Next, we sum up the count numbers of all documents for every result. Suppose that all results include \( l \) documents for the same query. For result \( r_i \), if the sum of count number of all documents in \( r_i \) is \( l \), which means that all documents in this result are totally different from the others and no overlap exists, we assign a weight of 1 to \( r_i \); if the sum is \( l \times n \), which means that all \( n \) results just include the same documents, we assign a weight of \( 1/n \) to \( r_i \); if the sum \( m \) is between \( l \) and \( l \times n \), we assign a weight of \( 1 - \frac{m-l}{l \times n} \) to \( r_i \).

The above weighting calculation can be carried out in every query and for that query only (Correlation Method 3). An alternative is keeping all historical weights and using their average for the forthcoming queries (Correlation Method 4).

Since the overlap rates of all \( n \) results (each with \( l \) documents) can be calculated by one scan of them, therefore the time complexity of the weights assignment process is \( O(l \times n) \). The data fusion process of these two methods takes
(O(l * n)) time as the linear combination method does.

The same method as in Correlation Methods 1 and 2 can be used for disparate performance among component results.

3.3 Correlation Methods 5 and 6

For \( n \) results \( r_i \) (\( 1 \leq i \leq n \)), correlation may exist among multiple results. In these two methods, we assign a weight for every possible combination of the \( n \) results. For \( n \) results, there are \( 2^n - 1 \) combinations in total. For example, if we have 3 results \( r_1, r_2, \) and \( r_3 \), then there are \( 2^3 - 1 = 7 \) combinations, which are \( \{r_1\}, \{r_2\}, \{r_3\}, \{r_1, r_2\}, \{r_1, r_3\}, \{r_2, r_3\}, \) and \( \{r_1, r_2, r_3\} \). We can use \( c_j \) (\( 1 \leq j \leq 2^n - 1 \)) to represent any of these combinations. There is a convenient way to decide the correspondence between a number \( j \) and the combination that \( c_j \) represents: we represent \( j \) in binary format with \( n \) digits. If there is a digit 1 in place \( i \) (counting from right to left), then \( r_i \) will be included in the combination; if 0 appears in place \( i \) (counting from right to left), then \( r_i \) will not be included in the combination. For three results, every \( c_j \) and its corresponding combination is listed as below:

1. \( c_1 \{r_1\} \) (binary format of 1 is 001)
2. \( c_2 \{r_2\} \) (binary format of 2 is 010)
3. \( c_3 \{r_1, r_2\} \) (binary format of 3 is 011)
4. \( c_4 \{r_3\} \) (binary format of 4 is 100)
5. \( c_5 \{r_1, r_3\} \) (binary format of 5 is 101)
6. \( c_6 \{r_2, r_3\} \) (binary format of 6 is 110)
7. \( c_7 \{r_1, r_2, r_3\} \) (binary format of 7 is 111)

The above combination representation is useful for merged documents as well. For every document \( d \) in the merged result, very often we need to indicate all the results in which the document is included. It is a combination of the results, and the subscript number \( j \) (\( 1 \leq j \leq 2^n - 1 \)) in \( c_j \) is referred to as the inclusion number of the merged document later in this paper. Besides, we define two functions here, which will be used later. The first one is \( \text{binary}(i, j) \), which checks if a 1 or 0 appears in place \( i \) of \( j \)'s binary format. If 1 appears in place \( i \) of \( j \)'s binary format, then \( \text{binary}(i, j) = 1 \); If 0 appears in place \( i \) of \( j \)'s binary format, then \( \text{binary}(i, j) = 0 \). This function is used to decide if a particular result is included in a particular combination. The second one is \( \text{count}(j) \), which counts the number of 1's in \( j \)'s binary format. This function is used to calculate the number of results included in a particular combination. For example, \( \text{binary}(1, 5) = 1; \text{binary}(1, 4) = 0; \text{count}(11) = 3 \).

We assign a weight \( w_j \) to every combination \( c_j \) by a group of results from training queries or the current query under processing. For these weights, the following condition
must be satisfied for every result \( r_i \) (\( 1 \leq i \leq n \)). This guarantees that all the weights are normalized. If we have 3 results, then we should guarantee: \( w_1 + w_3 + w_5 + w_7 = 1 \) (for result 1), \( w_2 + w_3 + w_6 + w_7 = 1 \) (for result 2), and \( w_4 + w_5 + w_6 + w_7 = 1 \) (for result 3).

With all these weights, we are able to fuse results. For any document \( d \), we assume that it obtains \( n \) scores \((s_1(d), s_2(d), \ldots, s_n(d))\) from \( n \) results. If \( d \) does not be included in \( r_i \), then we simply assign a score of 0 to \( s_i(d) \). Then we use the following formula to calculate the global score of \( d \):

\[
g_{s}(d) = \frac{\sum_{j=2}^{2^n-1} w_j \sum_{i=1}^{n} (s_i(d) \cdot \text{binary}(i, j))}{\text{count}(j)}
\]

Finally, let us discuss how to assign these weights by training or on the fly. Suppose we obtain \( n \) results, all of which include the same number \((l)\) of documents, from all component systems for a given query. We can use the following algorithm to assign weights to \( w_j \) (\( 1 \leq j \leq 2^n - 1 \)).

**Algorithm 1. Weights Assignment for Correlation Methods 5 and 6**

**Input:** \( n \) results from \( n \) different systems retrieving the same document collection for the same query. Every result includes \( l \) documents.

**Output:** \( w_j \) (\( 1 \leq j \leq 2^n - 1 \))

**Step 1.** Combine \( n \) results together into a single set. For every combined document, we indicate the results in which the document is included using a inclusion number \( j \) (\( 1 \leq j \leq 2^n - 1 \)).

**Step 2.** Initialise \( w \): assign 0 to every \( w_j \).

**Step 3.** For every element in the combined set, if its inclusion number is \( j \), then let \( w_j = w_j + 1 \).

**Step 4.** Normalizing \( w_j \): for every \( w_j \), let \( w_j = w_j/l \).

Again, we have two options here for the above weights assignment process: to carry it out in every query and use it for that query only (**Correlation Method 5**), or to keep all historical data and use their average for the forthcoming queries (**Correlation Method 6**).

The time complexity of correlation method 5 and 6 are \( O(l \cdot n + 2^n) \) for weights assignment, since it needs one scan of all results and all \( 2^n - 1 \) weights need to be assigned. For these two methods, the time complexity of the data fusion process is variable. One extreme situation is that all \( n \) results include the same collection of documents. Then we have \( l \) merged documents, and each of these documents has to use all \( 2^n - 1 \) weights. Therefore, in such a situation the time complexity is \( O(l \cdot 2^n) \). If there is no or very little overlapping documents, then they only take \( O(l \cdot n) \) time as the linear combination method.
<table>
<thead>
<tr>
<th>Group</th>
<th>Ave. MRR</th>
<th>Deviation of MRR</th>
<th>Ave. ( o_{rate} )</th>
<th>Deviation of ( o_{rate} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREC 6</td>
<td>0.7672</td>
<td>0.0719</td>
<td>0.3777</td>
<td>0.1309</td>
</tr>
<tr>
<td>TREC 7</td>
<td>0.8196</td>
<td>0.0247</td>
<td>0.4414</td>
<td>0.1142</td>
</tr>
<tr>
<td>TREC 8</td>
<td>0.8416</td>
<td>0.0416</td>
<td>0.4925</td>
<td>0.1274</td>
</tr>
<tr>
<td>TREC 9</td>
<td>0.6247</td>
<td>0.0504</td>
<td>0.5185</td>
<td>0.1542</td>
</tr>
<tr>
<td>TREC 2001</td>
<td>0.6637</td>
<td>0.0422</td>
<td>0.5209</td>
<td>0.1396</td>
</tr>
<tr>
<td>TREC 2002</td>
<td>0.4571</td>
<td>0.0138</td>
<td>0.5332</td>
<td>0.2023</td>
</tr>
</tbody>
</table>

In summary, all six correlation methods presented in the above are generalizations of CombSum. For correlation method 1-4, they become CombSum when all results obtain equal weights. For correlation method 5 and 6, the condition is that all combinations with two or more results have a weight of 0 while the combinations with only one result have a weight of 1. Correlation Method 5 and 6 are more precise than 4 other correlation methods since all types of possible correlation (e.g., correlation among more than two results) have been considered.

4 Experimental Setting & Results

We have used six groups of results, which were the top 10 results \(^1\) submitted to TREC 6 (ad-hoc track), 7 (ad-hoc track), 8 (ad-hoc track), 9 (web track), 2001 (web track), and 2002 (web track). MRR (Mean Reciprocal Rank), a different measure from precision and recall, which has not been used for the evaluation of data fusion, is used in the evaluation of our experimental results in support of high accuracy retrieval.

Some statistics of these six year group are presented in Table 1 (See also the Appendix for the titles and performances of all runs involved). We use "\( o_{rate} \)" to stand for overlap rate of all results. Among all these 6 year groups, TREC 8 is the best with an average of 0.8416 in performance, while TREC 2002 is the worst, whose average performance is only 0.4571.

Out of a total of 10 results, there are 45 pairs of different combinations in each group. We measured the overlap rates of all these pairs and averaged them. These averages are presented in the column "Average overlap rate" in Table 1. Considerable overlap exists in all cases. For all 6 groups, average overlap rates varies from 0.3773 to 0.5332, increasing monotonously year by year.

We used different number \( \{3, 4, 5, 6, 7, 8\} \) of results for each data fusion run. Out of 10 results, there are 120, 210, 252, 210, 120, and 45 different combinations for 3, 4, 5, 6, 7, and 8 results, respectively. We performed data fusion runs with all possible combinations in all 6 groups. Eight data fusion methods were

---

\(^1\)For convenience, all selected results include 1000 documents for each query.
Table 2: Summary of performance comparison of data fusion methods (3-8 results, 957 combinations in TREC 6, 7, 8, 9, 2001 and 2002, 50 queries for each combination)

<table>
<thead>
<tr>
<th>Group</th>
<th>MRR</th>
<th>P_{im,b}</th>
<th>P_{n,b}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>0.7457</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MNZ</td>
<td>0.7682</td>
<td>3.20%</td>
<td>73.45%</td>
</tr>
<tr>
<td>Sum</td>
<td>0.7689</td>
<td>3.34%</td>
<td>71.82%</td>
</tr>
<tr>
<td>Cor1</td>
<td>0.7707</td>
<td>3.53%</td>
<td>74.48%</td>
</tr>
<tr>
<td>Cor2</td>
<td>0.7708</td>
<td>3.51%</td>
<td>76.09%</td>
</tr>
<tr>
<td>Cor3</td>
<td>0.7704</td>
<td>3.54%</td>
<td>73.86%</td>
</tr>
<tr>
<td>Cor4</td>
<td>0.7698</td>
<td>3.42%</td>
<td>73.06%</td>
</tr>
<tr>
<td>Cor5</td>
<td>0.7712</td>
<td>3.57%</td>
<td>75.64%</td>
</tr>
<tr>
<td>Cor6</td>
<td>0.7712</td>
<td>3.60%</td>
<td>75.69%</td>
</tr>
</tbody>
</table>

used: CombSum (Sum), CombMNZ (MNZ), and Correlation Method 1-6 (Cor1, Cor2, ..., Cor6). The average performances of these data fusion methods are presented in Table 2 for all 6 year groups (all possible combinations of 3-8 results out of 10 results, 5742 in total). The figures in the "Performance" row indicate the performance (in MRR) of the fused result using a particular data fusion method as well as of the best result involved in the fusion, while the figure in the "P_{im,b}" row indicate the percentage of improvement on performance of the data fusion method over the best result involved, and the figures in the "P_{n,b}" row indicate the percentage of runs at which the fused result is better than the best result involved.

For all methods, their average performances are very close, and so are the percentages of improvement over the best result, which is between 3.20 and 3.60 for all methods. A two-tailed paired-samples t test was done to compare the mean of the best result and of the fused results. It shows that for all 8 fusion methods their performances are very significantly better than that of the best component result at a significance level of .000. We also compared the performances of all data fusion methods involved in the experiment. It shows that CombMNZ is worse than CombSum at a significance level of .016, while both CombSum and CombMNZ are worse than all six correlation methods at a significance level of .000. In six correlation methods, method 5 and 6 are better than the four other methods at a significance level of no more than .016 (many of them are .000), while the performance of correlation method 5 and 6 are not significantly different. All correlation methods are slightly better than CombSum on the percentage of runs which are better on "P_{n,b}" than the best component result. Also most of the combination methods (5 out of 6) outperform CombMNZ on this respect.

Comparing correlation method 1 and 2, 3 and 4, and 5 and 6, the experimental results suggest that assigning weights in both ways (on the fly for every query and training by many queries) are equally effective.
Table 3: Performance comparison of a group of data fusion methods (3-8 systems, 50 queries for each combination)

<table>
<thead>
<tr>
<th>Method</th>
<th>Measure</th>
<th>3 Sys</th>
<th>4 Sys</th>
<th>5 Sys</th>
<th>6 Sys</th>
<th>7 Sys</th>
<th>8 Sys</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNZ</td>
<td>MRR</td>
<td>0.7537</td>
<td>0.7628</td>
<td>0.7686</td>
<td>0.7728</td>
<td>0.7777</td>
<td>0.7824</td>
</tr>
<tr>
<td></td>
<td>P_{im}</td>
<td>2.98%</td>
<td>3.16%</td>
<td>3.19%</td>
<td>3.17%</td>
<td>3.39%</td>
<td>3.73%</td>
</tr>
<tr>
<td></td>
<td>P_{nb}</td>
<td>71.56%</td>
<td>74.64%</td>
<td>74.24%</td>
<td>73.78%</td>
<td>71.35%</td>
<td>72.46%</td>
</tr>
<tr>
<td>Sum</td>
<td>MRR</td>
<td>0.7541</td>
<td>0.7629</td>
<td>0.7691</td>
<td>0.7737</td>
<td>0.7768</td>
<td>0.7816</td>
</tr>
<tr>
<td></td>
<td>P_{im}</td>
<td>3.04%</td>
<td>3.24%</td>
<td>3.37%</td>
<td>3.42%</td>
<td>3.48%</td>
<td>3.66%</td>
</tr>
<tr>
<td></td>
<td>P_{nb}</td>
<td>70.34%</td>
<td>73.37%</td>
<td>72.59%</td>
<td>71.71%</td>
<td>69.40%</td>
<td>71.01%</td>
</tr>
<tr>
<td>Cor1</td>
<td>MRR</td>
<td>0.7548</td>
<td>0.7650</td>
<td>0.7715</td>
<td>0.7759</td>
<td>0.7813</td>
<td>0.7830</td>
</tr>
<tr>
<td></td>
<td>P_{im}</td>
<td>3.10%</td>
<td>3.43%</td>
<td>3.56%</td>
<td>3.62%</td>
<td>3.93%</td>
<td>3.68%</td>
</tr>
<tr>
<td></td>
<td>P_{nb}</td>
<td>71.21%</td>
<td>75.75%</td>
<td>76.22%</td>
<td>75.43%</td>
<td>72.31%</td>
<td>68.84%</td>
</tr>
<tr>
<td>Cor2</td>
<td>MRR</td>
<td>0.7559</td>
<td>0.7655</td>
<td>0.7713</td>
<td>0.7760</td>
<td>0.7803</td>
<td>0.7836</td>
</tr>
<tr>
<td></td>
<td>P_{im}</td>
<td>3.20%</td>
<td>3.48%</td>
<td>3.51%</td>
<td>3.56%</td>
<td>3.71%</td>
<td>3.78%</td>
</tr>
<tr>
<td></td>
<td>P_{nb}</td>
<td>73.61%</td>
<td>76.78%</td>
<td>77.81%</td>
<td>76.00%</td>
<td>74.12%</td>
<td>75.73%</td>
</tr>
<tr>
<td>Cor3</td>
<td>MRR</td>
<td>0.7546</td>
<td>0.7649</td>
<td>0.7707</td>
<td>0.7757</td>
<td>0.7808</td>
<td>0.7846</td>
</tr>
<tr>
<td></td>
<td>P_{im}</td>
<td>3.08%</td>
<td>3.45%</td>
<td>3.50%</td>
<td>3.60%</td>
<td>3.89%</td>
<td>4.08%</td>
</tr>
<tr>
<td></td>
<td>P_{nb}</td>
<td>70.94%</td>
<td>76.07%</td>
<td>75.20%</td>
<td>74.25%</td>
<td>71.35%</td>
<td>69.20%</td>
</tr>
<tr>
<td>Cor4</td>
<td>MRR</td>
<td>0.7550</td>
<td>0.7647</td>
<td>0.7707</td>
<td>0.7756</td>
<td>0.7799</td>
<td>0.7735</td>
</tr>
<tr>
<td></td>
<td>P_{im}</td>
<td>3.14%</td>
<td>3.44%</td>
<td>3.50%</td>
<td>3.60%</td>
<td>3.74%</td>
<td>1.76%</td>
</tr>
<tr>
<td></td>
<td>P_{nb}</td>
<td>71.40%</td>
<td>76.64%</td>
<td>75.43%</td>
<td>74.17%</td>
<td>71.90%</td>
<td>54.37%</td>
</tr>
<tr>
<td>Cor5</td>
<td>MRR</td>
<td>0.7552</td>
<td>0.7654</td>
<td>0.7721</td>
<td>0.7767</td>
<td>0.7811</td>
<td>0.7841</td>
</tr>
<tr>
<td></td>
<td>P_{im}</td>
<td>3.16%</td>
<td>3.48%</td>
<td>3.63%</td>
<td>3.68%</td>
<td>3.83%</td>
<td>3.89%</td>
</tr>
<tr>
<td></td>
<td>P_{nb}</td>
<td>71.90%</td>
<td>75.75%</td>
<td>77.87%</td>
<td>76.86%</td>
<td>74.52%</td>
<td>69.93%</td>
</tr>
<tr>
<td>Cor6</td>
<td>MRR</td>
<td>0.7555</td>
<td>0.7654</td>
<td>0.7721</td>
<td>0.7766</td>
<td>0.7811</td>
<td>0.7840</td>
</tr>
<tr>
<td></td>
<td>P_{im}</td>
<td>3.21%</td>
<td>3.49%</td>
<td>3.66%</td>
<td>3.68%</td>
<td>3.82%</td>
<td>3.80%</td>
</tr>
<tr>
<td></td>
<td>P_{nb}</td>
<td>72.84%</td>
<td>75.84%</td>
<td>77.35%</td>
<td>75.61%</td>
<td>75.93%</td>
<td>72.91%</td>
</tr>
</tbody>
</table>

Table 3 present the performances of these data fusion methods grouping with the same number of results. A common tendency for all these data fusion methods is: their performances increase slightly when we have more results for the fusion, so do the values of "P_{im,b}". The values of "P_{nb}" are between 54% and 78% in all cases. With 3-7 results, all six correlation methods outperform CombSum and CombMNZ. However, some of the correlation methods do not perform as well as CombSum and CombMNZ with 8 results.

If the correlations among all component results are equal or close to each other, then there is no or small difference between CombSum and correlation methods. If the correlations among all component results are quite different, then we can expect that the difference between CombSum and correlation methods becomes larger. We choose a half of all combinations which have stronger correlation then observe the performances of all these data fusion methods. The result is shown in Table 4. Compare Table 4 with Table 2, we find that most
Table 4: Summary of performance comparison of data fusion methods (3-8 results, 957 combinations in TREC 6, 7, 8, 9, 2001 and 2002, 50 queries for each combination)

<table>
<thead>
<tr>
<th>Group</th>
<th>MRR</th>
<th>P_{im,b}</th>
<th>P_{n,b}</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNZ</td>
<td>0.7662</td>
<td>2.73%</td>
<td>70.28%</td>
</tr>
<tr>
<td>Sum</td>
<td>0.7650</td>
<td>2.61%</td>
<td>68.46%</td>
</tr>
<tr>
<td>Cor1</td>
<td>0.7692</td>
<td>3.09%</td>
<td>72.90%</td>
</tr>
<tr>
<td>Cor2</td>
<td>0.7706</td>
<td>3.24%</td>
<td>74.76%</td>
</tr>
<tr>
<td>Cor3</td>
<td>0.7693</td>
<td>3.11%</td>
<td>73.03%</td>
</tr>
<tr>
<td>Cor4</td>
<td>0.7699</td>
<td>3.19%</td>
<td>73.79%</td>
</tr>
<tr>
<td>Cor5</td>
<td>0.7711</td>
<td>3.29%</td>
<td>75.19%</td>
</tr>
<tr>
<td>Cor6</td>
<td>0.7712</td>
<td>3.33%</td>
<td>75.19%</td>
</tr>
</tbody>
</table>

figures in Table 4 are smaller than their counterparts in Table 2. It demonstrates that correlation affects data fusion. However, the effect of correlation is not even to different data fusion methods. Since special treatment has been provided for all correlation data fusion methods, they are able to resist correlation more effectively than CpmSum and CombMNZ. In such a situation, all correlation methods are better than combSum and CombMNZ at a significance level of .000.

5 Some more observations

Next we present the experimental results based on year group, which are shown in Table 5 and 6. For all methods in all year groups, the average value of \( P_{n,b} \) is not negative, which shows that these data fusion methods are quite consistent in performance. In Table 5, the percentage of runs which are better than the best result varies from about 40% (Cor1, TREC9) to over 90% (several correlation methods in TREC 6 and TREC 2002).

In all six year groups, 10 systems in TREC 8 were the most accurate, with an average of 0.8196 on MRR. Even in such a situation, all data fusion methods perform very well, with an average of between 0.9249 and 0.9308 and all outperforming the best result by over 5%. Moreover, over 95% of the times all data fusion methods outperform the best component result. This suggests that data fusion can be effective with very high accuracy retrieval results as long as other factors of results are favourable.

We also notice that in two year groups, TREC 7 and TREC 9, all data fusion methods do not perform as well as they do in 4 other year groups on both measures, \( P_{im,b} \) and \( P_{n,b} \). It is interesting to investigate why this happens.

Let us look at TREC 7 first. In TREC 7, the best result is input.CLARIT98CLUS, whose average performance over 50 queries is 0.8885, 9.4% above the average performance (0.8119) of 9 other results. In such a situation, it is quite difficult
for data fusion methods to outperform the best result \textit{input.CLARIT98CLUS} when it is involved in the fusion. Thus, we divide all combinations into two groups: with and without \textit{input.CLARIT98CLUS}, which is presented in Table 7. The group with \textit{input.CLARIT98CLUS} (Group 1) performs much worse than the group without \textit{input.CLARIT98CLUS} (Group 2), especially on \textit{"Pn_b"}. In Group 1, all six correlation methods perform better than CombSum, while the difference between CombSum and correlation methods is smaller in Group 2. This is because \textit{input.CLARIT98CLUS} is very different from the others. The average overlap rate among all possible pairs is 0.4414, while that between \textit{input.CLARIT98CLUS} and 9 other results is only 0.2780.

Next let us look at TREC 9. The top two systems are \textit{input.NEnm} and \textit{input.NEnmLpas}. Their overlap rate over all 50 queries are 1! Their average performances are 0.7133 and 0.7129, respectively, which are 18.4\% and 18.3\% above the average performance (0.6027) of 8 other results. Therefore, all data fusion methods do not perform as well as the best component result when both (Group 1 in Table 8) or either (Group 2 in Table 8) of them are involved in the fusion. When neither of them are involved (Group 3 in Table 8), all data fusion methods perform very well.

Before we finish this section, we would like to make some comments about CombMNZ. Both CombSum and Comb-MNZ were presented by Fox and his colleagues in (Fox et al., 1993; Fox & Shaw, 1994). Actually, comparing the score calculating methods of CombSum and CombMNZ, which are $\sum_{i=1}^{n} score_i$ and $\sum_{i=1}^{n} score_i \times n$, respectively, they bear some similarity. The difference between them is an extra factor $n$ for CombMNZ. For those documents retrieved by

<table>
<thead>
<tr>
<th>Method</th>
<th>TREC6</th>
<th>TREC7</th>
<th>TREC8</th>
<th>TREC9</th>
<th>TREC01</th>
<th>TREC02</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNZ</td>
<td>0.8886</td>
<td>0.8612</td>
<td>0.9249</td>
<td>0.6962</td>
<td>0.7341</td>
<td>0.5043</td>
</tr>
<tr>
<td></td>
<td>(4.18%)</td>
<td>(0.45%)</td>
<td>(5.14%)</td>
<td>(0.20%)</td>
<td>(2.63%)</td>
<td>(6.59%)</td>
</tr>
<tr>
<td>Sum</td>
<td>0.8820</td>
<td>0.8586</td>
<td>0.9290</td>
<td>0.6964</td>
<td>0.7364</td>
<td>0.5107</td>
</tr>
<tr>
<td></td>
<td>(3.41%)</td>
<td>(0.14%)</td>
<td>(5.61%)</td>
<td>(0.24%)</td>
<td>(2.69%)</td>
<td>(7.94%)</td>
</tr>
<tr>
<td>Cor1</td>
<td>0.8908</td>
<td>0.8642</td>
<td>0.9302</td>
<td>0.6960</td>
<td>0.7372</td>
<td>0.5060</td>
</tr>
<tr>
<td></td>
<td>(4.45%)</td>
<td>(0.79%)</td>
<td>(5.75%)</td>
<td>(0.18%)</td>
<td>(3.08%)</td>
<td>(6.94%)</td>
</tr>
<tr>
<td>Cor2</td>
<td>0.8903</td>
<td>0.8633</td>
<td>0.9308</td>
<td>0.6983</td>
<td>0.7394</td>
<td>0.5029</td>
</tr>
<tr>
<td></td>
<td>(4.39%)</td>
<td>(0.68%)</td>
<td>(5.82%)</td>
<td>(0.52%)</td>
<td>(3.38%)</td>
<td>(6.29%)</td>
</tr>
<tr>
<td>Cor3</td>
<td>0.8886</td>
<td>0.8633</td>
<td>0.9295</td>
<td>0.6965</td>
<td>0.7357</td>
<td>0.5088</td>
</tr>
<tr>
<td></td>
<td>(4.19%)</td>
<td>(0.68%)</td>
<td>(5.67%)</td>
<td>(0.26%)</td>
<td>(2.87%)</td>
<td>(7.54%)</td>
</tr>
<tr>
<td>Cor4</td>
<td>0.8866</td>
<td>0.8613</td>
<td>0.9303</td>
<td>0.6977</td>
<td>0.7374</td>
<td>0.5052</td>
</tr>
<tr>
<td></td>
<td>(3.95%)</td>
<td>(0.48%)</td>
<td>(5.76%)</td>
<td>(0.43%)</td>
<td>(3.10%)</td>
<td>(6.77%)</td>
</tr>
<tr>
<td>Cor5</td>
<td>0.8924</td>
<td>0.8651</td>
<td>0.9294</td>
<td>0.6954</td>
<td>0.7398</td>
<td>0.5053</td>
</tr>
<tr>
<td></td>
<td>(4.64%)</td>
<td>(0.89%)</td>
<td>(5.66%)</td>
<td>(0.00%)</td>
<td>(3.44%)</td>
<td>(6.80%)</td>
</tr>
<tr>
<td>Cor6</td>
<td>0.8890</td>
<td>0.8632</td>
<td>0.9305</td>
<td>0.6980</td>
<td>0.7418</td>
<td>0.5049</td>
</tr>
<tr>
<td></td>
<td>(4.23%)</td>
<td>(0.67%)</td>
<td>(5.79%)</td>
<td>(0.48%)</td>
<td>(3.72%)</td>
<td>(6.70%)</td>
</tr>
</tbody>
</table>
Table 6: Comparison of six groups of results (percentage of results which are better than the best result)

<table>
<thead>
<tr>
<th>Method</th>
<th>TREC6</th>
<th>TREC7</th>
<th>TREC8</th>
<th>TREC9</th>
<th>TREC01</th>
<th>TREC02</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNZ</td>
<td>88.61%</td>
<td>52.35%</td>
<td>95.72%</td>
<td>44.97%</td>
<td>71.79%</td>
<td>87.25%</td>
</tr>
<tr>
<td>Sum</td>
<td>82.34%</td>
<td>47.23%</td>
<td>96.66%</td>
<td>42.90%</td>
<td>70.64%</td>
<td>91.12%</td>
</tr>
<tr>
<td>Cor1</td>
<td>90.44%</td>
<td>55.60%</td>
<td>96.68%</td>
<td>39.58%</td>
<td>74.87%</td>
<td>89.72%</td>
</tr>
<tr>
<td>Cor2</td>
<td>89.86%</td>
<td>56.01%</td>
<td>97.07%</td>
<td>46.31%</td>
<td>77.53%</td>
<td>89.76%</td>
</tr>
<tr>
<td>Cor3</td>
<td>88.05%</td>
<td>53.81%</td>
<td>96.68%</td>
<td>40.52%</td>
<td>73.11%</td>
<td>90.96%</td>
</tr>
<tr>
<td>Cor4</td>
<td>87.12%</td>
<td>50.60%</td>
<td>96.78%</td>
<td>43.32%</td>
<td>73.83%</td>
<td>89.72%</td>
</tr>
<tr>
<td>Cor5</td>
<td>91.90%</td>
<td>57.45%</td>
<td>96.58%</td>
<td>39.59%</td>
<td>78.60%</td>
<td>89.72%</td>
</tr>
<tr>
<td>Cor6</td>
<td>90.18%</td>
<td>54.54%</td>
<td>97.39%</td>
<td>40.42%</td>
<td>80.36%</td>
<td>91.22%</td>
</tr>
</tbody>
</table>

Table 7: Performance of a group of data fusion methods in TREC 7 (with input.CLARIT98CLUS - group 1, without input.CLARIT98CLUS - group 2)

<table>
<thead>
<tr>
<th>Method</th>
<th>Group 1</th>
<th>Group 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P_im_b</td>
<td>P_n_b</td>
</tr>
<tr>
<td>MNZ</td>
<td>-0.83%</td>
<td>26.28%</td>
</tr>
<tr>
<td>Sum</td>
<td>-1.20%</td>
<td>19.72%</td>
</tr>
<tr>
<td>Cor1</td>
<td>-0.47%</td>
<td>30.69%</td>
</tr>
<tr>
<td>Cor2</td>
<td>-0.55%</td>
<td>29.47%</td>
</tr>
<tr>
<td>Cor3</td>
<td>-0.54%</td>
<td>29.47%</td>
</tr>
<tr>
<td>Cor4</td>
<td>-0.74%</td>
<td>25.00%</td>
</tr>
<tr>
<td>Cor5</td>
<td>-0.40%</td>
<td>32.93%</td>
</tr>
<tr>
<td>Cor6</td>
<td>-0.51%</td>
<td>31.10%</td>
</tr>
</tbody>
</table>

When documents are retrieved by the same number of component systems, their relative ranking positions in the merged results are always the same for both CombSum and CombMNZ. Suppose \(d_1\) and \(d_2\) are two documents retrieved by the same number of component systems, then we can conclude: if \(d_1\)'s ranking is higher than \(d_2\)'s ranking in the merged result with CombSum, then \(d_1\)'s ranking is higher than \(d_2\)'s ranking in the merged result with CombMNZ; if \(d_1\)'s ranking is lower than \(d_2\)'s ranking in the merged result with CombSum, then \(d_1\)'s ranking is lower than \(d_2\)'s ranking in the merged result with CombMNZ. the conclusion is also true the other way round. When documents are retrieved by different number of component systems, CombSum and CombMNZ will rank them in different ways. Generally speaking, CombMNZ favours those documents which are retrieved by more component systems than CombSum does. For example, suppose we have two documents \(d_1\) and \(d_2\). \(d_1\) is retrieved by 3 component systems with a set of scores \(\{0.9, 0.8, 0.9\}\), while \(d_2\) is retrieved by 5 systems with a set of scores \(\{0.3, 0.2, 0.5, 0.3, 0.4\}\). With CombSum, \(d_1\) will be ranked ahead of \(d_2\); while with CombMNZ, \(d_2\) will be ranked ahead of \(d_1\). With such an observation, we
8 Conclusions

We conclude that in general, the data fusion technique is effective on performance improvement with high accuracy retrieval, because of the following observations:

- For all six groups of results, all eight data fusion algorithms involved outperform the best system by over 3% on average;
- For all six groups of results, over 70% of the times the fused result generated by any of these methods is better than the result from the best system;
- For the top ten results submitted to TREC 8, though their average performance in MRR is as high as 0.8416, all eight data fusion methods outperform the best system by over 5%.

Another conclusion is that the six correlation methods presented in this paper are effective on eliminating the harmful effect of uneven correlation among component results, because of the following observations in the experiment:

<table>
<thead>
<tr>
<th>Method</th>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P_{im,b}</td>
<td>P_{n,b}</td>
<td>P_{im,b}</td>
</tr>
<tr>
<td>MNZ</td>
<td>-0.75%</td>
<td>35.37%</td>
<td>-1.66%</td>
</tr>
<tr>
<td>Sum</td>
<td>-1.08%</td>
<td>30.08%</td>
<td>-1.64%</td>
</tr>
<tr>
<td>Cor1</td>
<td>-1.10%</td>
<td>31.71%</td>
<td>-1.73%</td>
</tr>
<tr>
<td>Cor2</td>
<td>-1.06%</td>
<td>30.49%</td>
<td>-1.70%</td>
</tr>
<tr>
<td>Cor3</td>
<td>-1.06%</td>
<td>30.49%</td>
<td>-1.71%</td>
</tr>
<tr>
<td>Cor4</td>
<td>-0.98%</td>
<td>34.15%</td>
<td>-1.49%</td>
</tr>
<tr>
<td>Cor5</td>
<td>-1.40%</td>
<td>22.76%</td>
<td>-1.67%</td>
</tr>
<tr>
<td>Cor6</td>
<td>-1.08%</td>
<td>28.05%</td>
<td>-1.36%</td>
</tr>
</tbody>
</table>
• All six correlation methods achieve higher MRR values than CombSum and CombMNZ. Though the differences between all methods are small, they are different at a significance level of .000;

• All six correlation methods are better than CombSum on the percentage of runs which are better than the best system; five of all six correlation methods are better than CombMNZ on that.

Therefore, these correlation methods are very likely to obtain better results than the well-known data fusion methods CombSum and CombMNZ. In addition, correlation methods 1-4 can be expanded to deal with divergent performance, with a compound weight to reflect both performance and correlation.

References

Nuray, R., & Can, F. Automatic ranking of retrieval systems using fusion data. Information Processing & Management, in press.

Wu, S., & McClean, S. Modelling rank-probability of relevance relationship in resultant document list for data fusion, submitted for publication.

Wu, S., & McClean, S. Performance prediction of data fusion for information retrieval. Information Processing & Management, in press.


Appendix: Systems in Every Year Group

In every year group, 10 best submission runs are selected. The figures in parentheses are the average performance of that run in MRR over 50 queries.

**TREC 6**
input.uwmt6a0 (0.8770), input.CLAUG (0.8319),
input.CLREL (0.8331), input.anu6min1 (0.8036),
input.LNmShort (0.8316), input.geru1 (0.7377),
input.Mercure1 (0.7044), input.anu6alo1 (0.6952),
input.Cor6A3cll (0.6799), input.Brkly22 (0.6776)

**TREC 7**
input.CLARIT98CLUS (0.8885), input.acsys7mi (0.8324),
input.att98atdc (0.8196), input.Brkly26 (0.8173),
input.uoftimgr (0.8129), input.tno7cbm25 (0.8110),
input.tno7exp1 (0.8081), input.tno7tw4 (0.8058),
input.ok7ax (0.8041), input.INQ502 (0.7968)

**TREC 8**
input.manexT3D1N (0.8760), input.CL99SD (0.8588),
input.CL99SDopt1 (0.8853), input.CL99SDopt2 (0.8726),
input.CL99XT (0.8527), input.CL99XTopt (0.8757),
input.GE8MTD2 (0.8546), input.Flab8atdn (0.7910),
input.fub99tf (0.7754), input.apl8p (0.7738)

**TREC 9**
input.NEmm (0.7133), input.NEmmLpas (0.7129),
input.iit00td (0.6465), input.iit00tde (0.6347),
input.Sab9web3 (0.6232), input.tnout9fl (0.6029),
input.Sab9web2 (0.5862), input.apl9all (0.5814),
input.Sab9web4 (0.5781), input.jscht9wll2 (0.5681)

**TREC 2001**
input.ok10wtn0 (0.7337), input.ok10wtn1 (0.7059),
input.kuadhoc2001 (0.6916), input.flabxtd (0.6886),
input.hum01tdlx (0.6736), input.flabxtdn (0.6627),
input.UniNEn7d (0.6625), input.ricAP (0.6089),
input.ricMS (0.6064), input.ricMM (0.6051)

**TREC 2002**
input.icttd1 (0.4350), input.Mercah (0.4682),
input.pltr02wt2 (0.4467), input.thutd1 (0.4630),
input.thutd2 (0.4664), input.thutd3 (0.4763),
input.thutd4 (0.4613), input.thutd5 (0.4447),
input.uog03ctadq (0.4580), input.uog05tad (0.4726)