Data & Knowledge Engineering 71 (2012) 114–126

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Linear combination of component results in information retrieval

Shengli Wu [*]

School of Computing and Mathematics, University of Ulster, Newtownabbey, UK

ARTICLE INFO

Article history:
Received 26 July 2010
Received in revised form 26 August 2011
Accepted 29 August 2011
Available online 10 September 2011

Keywords:
Information retrieval
Data fusion
Meta-search
The linear combination method
Multiple linear regression
Logistic regression
Model

ABSTRACT

In information retrieval, data fusion (also known as meta-search) has been investigated by many researchers. Previous investigation and experimentation demonstrate that the linear combination method is an effective data fusion method for combining multiple information retrieval results. One advantage is its flexibility, since different weights can be assigned to different component systems so as to obtain better fusion results. The key issue is how to assign good weights to all the component retrieval systems involved. Surprisingly, research in this field is limited and it is still an open question. In this paper, we use the multiple linear regression technique with estimated relevance scores and judged scores to obtain suitable weights. Although the multiple linear regression technique is not new, the way of using it in this paper has never been attempted before for the data fusion problem in information retrieval. Our experiments with five groups of runs submitted to TREC show that the linear combination method with such a weighting strategy steadily outperforms the best component system and other data fusion methods including CombSum, CombMNZ, PosFuse, MAPFuse, SegFuse, and the linear combination method with performance level/performance square weighting schemes by large margins.

1. Introduction

In information retrieval, the data fusion technique has been investigated by many researchers and quite a few data fusion methods such as CombSum [9,10], CombMNZ [9,10], Borda count [2], Condorcet fusion [19], the correlation method [30], Markov chain-based methods [5,21], the linear combination method [3,26,27,29], the multiple criteria approach [8], and others [6,14,16,23,33], have been presented and investigated. Previous research demonstrates that, generally speaking, data fusion is an effective technique for obtaining better results.

Data fusion methods for information retrieval can be divided into two categories: score-based methods and rank-based methods. These two different types of methods apply to different situations. If some component systems only provide a ranked list of documents as the result of a query, then rank-based methods can be applied. If each component system provides scores for all retrieved documents, then score-based methods can be applied. In those aforementioned methods, CombSum [9,10], CombMNZ [9,10], the correlation method [30], the linear combination method [3,26,27,29] are score-based methods, while Condorcet fusion [19], Markov chain-based methods [5,21], and the multiple criteria approach [8] are rank-based methods.

Among those score-based data fusion methods, the linear combination method is very flexible since different weights can be used for different component systems. It is especially beneficial when component systems involved vary considerably in performance and dissimilarity between each other. However, how to assign suitable weights to those systems involved is a crucial issue that needs to be carefully investigated.

Some previous investigation (e.g., [2,24]) took a simple performance level policy. Under this policy, all component information retrieval systems are evaluated over a group of training queries using the same document collection. If system *a*'s average

* Tel.: +44 2890366585; fax: +44 2890366216.
E-mail address: s.wu1@ulster.ac.uk.

0169-023X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2011.08.003

performance is $p(a)$, by a given metric (e.g., mean average precision), then we take $p(a)$ as a's weight for the linear combination method. Some alternatives to the simple performance level weighting scheme were investigated in [29]. On the other hand, both system performance and dissimilarities between systems were considered in [30] to determine all systems' weights. Although the linear combination method with these weighting schemes usually outperforms CombSum and CombMNZ — the two commonly used data fusion methods that assign equal weights to all component systems, it is very likely that there is still room for further improvement since all these weighting schemes are heuristic.

In this paper we discuss a new approach, multiple linear regression, to training system weights for data fusion. Our experiments demonstrate that this approach is more effective than other approaches. To our knowledge, multiple linear regression has not been used for such a purpose before, though it has been used in various other tasks in information retrieval.[1]

Another issue is how to obtain reliable scores for those retrieved documents. Sometimes scores are provided by component retrieval systems, but usually those raw scores from different retrieval systems are not comparable and some kind of score normalization is needed before fusing them. An alternative is to convert ranking information into scores if raw scores are unavailable or unreliable. Different models [1,2,4,17,20] have been investigated for such a purpose. In this piece of work, we compared several approaches and found that the logistic regression model and the cubic regression model could provide reliable scores for the linear combination methods. The binary logistic model has been investigated in the distributed information retrieval context [4,20], in which no overlap or partial overlap exists between the document collections for component retrieval systems.

The rest of this paper is organized as follows: in Section 2 we review some related work on data fusion. Then in Section 3 we discuss the linear combination method for data fusion and especially the two major issues involved. Section 4 presents the experimental settings and results for evaluating a group of data fusion methods including the one presented in this paper. Section 5 presents the experimental settings and results for evaluating a group of methods of normalizing scores and generating scores from ranking information. Section 6 is the conclusion.

2. Related work

Information retrieval has been widely used in various application areas such as Web search engines [12], digital libraries [6], databases [7], and many others. However, how to further improve retrieval performance is a demanding issue.

In these days, people find that the data fusion technology is very useful for performance improvement in various kinds of applications such as image [31], multiple sensor systems [13], databases [32], information retrieval [11,27], and so on.

In the following we mainly review some previous work on the linear combination method and score normalization methods for data fusion in information retrieval.

It was probably Thompson who reported the earliest work on the linear combination data fusion method in [24]. The linear combination method was used to fuse results submitted to TREC 1. It was found that the combined results, weighted by performance level, performed no better than a combination using uniform weights (CombSum). Weighted Borda count [2] is a variation of performance level weighting, in which all ranked documents are assigned scores corresponding to their respective Borda counts, and then the linear combination method with the performance level weighting is used for fusion.

Bartell and Cottrell [3] used conjugate gradient, a numerical optimisation method, to find good weights of different systems. Because the method is time-consuming, only 2 to 3 component systems and the top 15 documents returned from each system for a given query were used in their investigation.

Vogt and Cottrell [26,27] analysed the performance of the linear combination method by linear regression. Note that in their work, the linear regression technique is used for a different purpose (performance prediction) from the work reported in this paper (weights assignment). In their experiments, they used all possible pairs of 61 systems submitted to the TREC 5 ad-hoc track. Another numerical optimisation method, the golden section search, was used to search for best system weights. Due to the nature of the golden section search, only two component systems can be involved for each fusion case.

Effort has also gone into finding cost-effective, high-performance weighting schemes. In a piece of work done by Wu et al. [29], their experiment shows that, using the power function of performance with a power value between 2 and 6 can lead to slightly better fusion results than performance level weighting. On the other hand, in Wu and McClean's work [30], both system performance and dissimilarities between systems were considered. In their weighting scheme, any system's weight is a compound weight that is the product of its performance weight and its dissimilarity weight. Here the performance weight is also defined as the average performance of the system over a group of training queries as the performance level weighting scheme does, while the dissimilarity weight is defined as the average dissimilarity between the system in question and all other component systems over a group of training queries.

In [28], a geometric probabilistic framework is used to formally describe data fusion, in which each component result returned from an information retrieval system for a given query is represented as a point in a multiple dimensional space. Then all the component results and data fusion results can be explained using geometrical principles. In such a framework, it becomes clear why quite often data fusion methods such as the averaging-score method (CombSum) and the linear combination method can bring improvements in effectiveness and accordingly what the favourable conditions are for data fusion algorithms to achieve better results.

[1] In machine learning, multiple linear regression is used in many places. For example, for the classifier ensemble problem, researchers find that stacking with multi-response linear regression is a very successful meta-model [22,25].

A related problem is how to obtain reliable scores for retrieved documents. This problem is not trivial by any means. Sometimes component information retrieval systems provide scores for their retrieved documents. However, those raw scores from different retrieval systems may not be comparable. Some kind of normalization is required for those raw scores. One common linear score normalization method is: for any list of scores (associated with a ranked list of documents) for a given topic or query, we map the highest score into 1, the lowest score into 0, and any other scores into a value between 0 and 1 by using the formula

$$n\_score = \frac{(raw\_s - min\_s)}{(max\_s - min\_s)} \tag{1}$$

Here *max_s* is the highest score, *min_s* is the lowest score, *raw_s* is the score to be normalized and *n_s* is the normalized score of *raw_s*. This normalization method, referred to as 0–1 linear score normalization method later in this paper, was used by Lee [15] and others in their experiments.

Two other linear score normalization methods, SUM and ZMUV (Zero-Mean and Unit-Variance), were investigated by Montague and Aslam [18]. 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. ZMUV is known as z-score in statistics.

If only a ranked list of documents is provided without any scoring information, then we cannot use the linear combination method directly. A common way of dealing with this is to assign a given score to documents at a certain rank. For example, Borda count [2] works like this: for a ranked list of t documents, the first document in the list is given a score of t, the second document in the list is given a score of $t-1$, …, the last document in the list is given a score of 1. Thus all documents are assigned corresponding scores based on their rank positions and CombSum or the linear combination method can be applied accordingly.

Scores can also be converted from ranking information with some training data. In MAPFuse [16], a document at rank t is assigned a score of v_{map}/t, where v_{map} is the MAP value of the system in question over a group of training queries using a training document collection. PosFuse [16] also use a group of training queries and a training document collection. For the ranked list of documents from each component system, the posterior probabilities of documents at each given rank are observed and averaged over a group of queries. Then the averaged posterior probability is used as relevance scores for the document at that rank. In SegFuse [23], scores are generated using a mixture of normalized raw scores (s_{n-rs}, raw scores are provided by the information retrieval system) and ranking-related relevance scores (s_p, from ranking-related posterior probabilities). For s_{n-rs}, the 0–1 linear score normalization method is used to normalize raw scores. For obtaining s_p, SegFuse also needs a group of training queries and a document collection. However, it takes a slightly different approach from PosFuse. Instead of calculating posterior probabilities of documents at each given rank, SegFuse divides the whole lists of documents into segments of different size ($size_k = (10 * 2^{k-1} - 5)$). In each segment, documents are assigned equal relevance scores. The final score of a document is given by $s_p(1 + s_{n-rs})$. For the fusion process, the aforementioned three methods just use the same method as CombSum does.

Some other non-linear methods have also been discussed in [1,4,17,20]. In [17], an exponential distribution for the set of non-relevant documents and a normal distribution for the set of relevant documents were used, and then a mixture model was defined to fit the real score distribution. A signal-to-noise approach was investigated in [1]. The logistic regression model was investigated in [4,20] in the environment of distributed information retrieval, in which the merged results are retrieved from different document collections. Such a scenario is a little different from the one for data fusion. However, score normalization remains the same sub-problem in both situations.

Although there is quite a large amount of work done on the data fusion issue in information retrieval, how to assign suitable weights for the linear combination method is still an open question. On the one hand, some methods such as performance level weighting scheme and its variations are efficient, but they are heuristic in nature and they are not very effective. On the other hand, some numerical optimisation methods such as golden section search and conjugate gradient are able to find very effective weights if enough time is taken. However, one significant problem with these optimisation methods is they are very time-consuming. It is almost impossible to use them in practice when a relatively large number of component systems are involved or in a dynamic situation when weights need to be recalculated quite frequently. The proposed method in this paper is much more effective than the performance level weighting scheme and its variations, and is much more efficient than the optimisation methods. The time needed for this approach should be affordable for many applications.[2]

3. The linear combination method

Suppose we have n information retrieval systems $ir_1, ir_2, \ldots, ir_n$, and for a given query q, each of them provides a result r_i. Each r_i comprises a ranked list of documents with associated scores. The linear combination method uses the following formula to calculate score for every document d:

$$M(d,q) = \sum_{i=1}^{n} \beta_i * s_i(d,q) \tag{2}$$

[2] For example, we used the statistical software R for the multiple linear regression on a personal computer with Intel Duo core E8500 3.16GHZ CPU and 2 GB of RAM. For a group of 10,000 data records and 20 variables (component systems), the time needed for the multiple linear regression to obtain weights is 0.15 s.

Here $s_i(d, q)$ is the normalized score of document d in result r_i, β_i is the weight assigned to system ir_i, $M(d, q)$ is the calculated score of d for q. All the documents can be ranked according to their calculated scores.

If we have no knowledge of the component systems, then an equal weight for every component system is a good policy. If we have some knowledge of those retrieval systems (e.g., evaluating performance by using some training data), then we are able to take a more profitable policy: good systems are assigned heavy weights and poor systems are assigned light weights [24,29]. On the other hand, if some systems produce more similar results than the others, then those systems should be assigned lighter weights than the others for better fusion results [30]. The crucial issue for the success of the linear combination method is if we can obtain appropriate weights for all component systems in an efficient manner.

Suppose there are m queries, n information retrieval systems, and r documents in a document collection D. For each query q^i, all information retrieval systems provide relevance scores to all the documents in the collection, therefore, we have $(s^i_{1k}, s^i_{2k}, \ldots, s^i_{nk}, y^i_k)$ for $i = (1, 2, \ldots, m)$, $k = (1, 2, \ldots, r)$. Here s^i_{jk} stands for the score assigned by retrieval system ir_j to document d_k for query q^i; y^i_k is the judged relevance score of d_k for query q^i. If binary relevance judgment is used, then it is 1 for relevant documents and 0 otherwise.

Now we want to estimate

$$Y = \left\{ y^i_k; i = (1, 2, \ldots, m), k = (1, 2, \ldots, r) \right\}$$

by a linear combination of scores from all component systems. We use the least squares estimates here, which for the βs are the values $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \ldots,$ and $\hat{\beta}_n$ for which the quantity

$$q = \sum_{i=1}^{m} \sum_{k=1}^{r} \left[y^i_k - \left(\hat{\beta}_0 + \hat{\beta}_1 s^i_{1k} + \hat{\beta}_2 s^i_{2k} + \ldots + \hat{\beta}_n s^i_{nk} \right) \right]^2$$

is a minimum. This is partly a matter of mathematical convenience and partly due to the fact that many relationships are actually of this form or can be approximated closely by linear equations. $\beta_0, \beta_1, \beta_2, \ldots,$ and β_n, the multiple linear regression coefficients, are numerical constants that can be determined from observed data.

In the least squares sense the coefficients obtained by multiple linear regression can bring us the optimum fusion results by the linear combination method, since they can be used to make the most accurate estimation of the relevance scores of all the documents to all the queries as a whole. In a sense this technique for fusion performance improvement is general since it is not directly associated with any special metrics such as average precision or recall-level precision. Theoretically, this approach should work well for any reasonably defined ranking-based metrics such as average precision or recall-level precision, since more accurately estimated scores for all the documents are able for us to obtain better rankings of those documents.

In the linear combination method, those multiple linear regression coefficients, $\beta_1, \beta_2, \ldots,$ and β_n, can be used as weights for $ir_1, ir_2, \ldots, ir_n$, respectively. We do not need to use β_0 to calculate scores, since the relative ranking positions of all the documents are not affected if a constant β_0 is added to all the scores.

Let us take an example to illustrate how to obtain the multiple regression coefficients from some data. Suppose we have a collection of 4 documents, 3 information retrieval systems, and 2 queries. For each query, each information retrieval system assigns a score to each document in the collection. Binary relevance judgment is used. The scoring information and relevance judgment of all documents are shown as follows.

Query	Doc	ir_1	ir_2	ir_3	Relevance
q_1	d_1	0.50	0.30	0.80	1
q_1	d_2	0.60	0.70	0.40	1
q_1	d_3	0.10	0.80	0.40	0
q_1	d_4	0.20	0.30	0.10	0
q_2	d_1	0.30	0.40	0.80	1
q_2	d_2	0.20	0.50	0.10	0
q_2	d_3	0.30	0.40	0.40	0
q_2	d_4	0.30	0.50	0.50	1

By setting ir_1, ir_2, and ir_3 as independent variables, and *relevance* as the dependent variable, we use R[3] with the "Rcmdr" package to obtain the multiple linear regression coefficients. For this example, we obtain $\beta_1 = 0.498$, $\beta_2 = 0.061$, and $\beta_3 = 1.979$ for ir_1, ir_2, and ir_3, respectively.

One related issue that also needs to be considered for using the linear combination method is how to obtain reliable scores. That is, for every document retrieved by a retrieval system, it should have a reliable score that indicates the probability of relevance of that document to the query.

In this research, we mainly use the binary logistic regression model [4] to estimate the relation between ranking position of documents and their probabilities of being relevant, because we consider it to be a good and reliable method. However, some other score normalization methods are also investigated and some experiments are conducted to compare them. See Section 5 for more details.

[3] R is a free statistical software. Its web site is located at http://www.r-project.org/.

The logistic model can be expressed by the following equation

$$probability(t) = \frac{e^{a+b*t}}{1+e^{a+b*t}} = \frac{1}{1+e^{-a-b*t}} \tag{3}$$

In Eq. (3), *probability*(t) is the probability of the document at rank t being relevant to the query. a and b are two parameters. As in [4], we use $ln(t)$ to replace t in Eq. (3). According to [4], the rationale of using the logarithm of rank instead of the rank itself is based on the consideration that retrieval schemes generally display documents in a descending order (from more relevant to less relevant) and the probability of relevance changes systematically with the rank order (or the serial order). A difference of 10 in ranks seems to be more significant between the ranks 20 and 30 than between 990 and 1000. These last ranks contain such a small number of relevant documents that it might be appropriate to ignore the difference between 990 and 1000. Therefore, we have

$$probability(t) = \frac{e^{a+b*ln(t)}}{1+e^{a+b*ln(t)}} = \frac{1}{1+e^{-a-b*ln(t)}} \tag{4}$$

Usually, the logistic function is a S-shape curve. The function in Eq. (4) is the composition of a logarithm function and a logistic function, the curve is somewhat different from that of a pure logistic function. Fig. 1 is an example of the function composition with $b<0$. The curve decreases monotonously when rank increases. When $t=1$, $\ln(t)=0$. The highest point of the curve is totally decided by a. When t is larger, b's effect on the curve becomes more significant. The composition of a logarithm function and a logistic function is still referred to as the logistic function in the rest of this paper.

Suppose we have a group of 6 results from an information retrieval system, all of which are evaluated by human judgment. $result_1$={1, 1, 0, 0, 1, 0, 0, 0 }, $result_2$={0, 1, 0, 0, 0, 1, 0, 0 }, $result_3$={1, 0, 1, 1, 0, 0, 0, 0 }, $result_4$={1, 0, 1, 0, 1, 1, 0, 0 }, $result_5$={1, 1, 1, 0, 0, 0, 1, 0 }, $result_6$={1, 0, 0, 1, 0, 0, 0, 0 }. Here in each result there are 8 documents. “1” represents a relevant one and “0” represents an irrelevant one. For example, in $result_1$, the first, second, and fifth documents are relevant, while all the others are irrelevant. We can run binary logistic regression by using ln(rank) as the independent variable and score (probability of relevance) as the dependent variable. Then we can obtain the values of the coefficients $a=1.512$, and $b=-1.585$.

4. Data fusion experiments

In this section we present our experiment settings and results with five groups of runs submitted to TREC. Their information is summarized in Table 1.

We only use those runs that include 1000 documents for every query. In each year group, a few runs include fewer than 1000 documents for some queries. Removing those runs with fewer documents provides us a homogeneous environment for the investigation.

In some runs, the raw scores assigned to documents are not reasonable. For example, in anu5aut1, anu5aut2, ibmgd2, and ibmge2 in TREC 5 and cirtrc82 in TREC 8, the same scores are assigned to all 1000 documents for some queries; in uwgcx0 of TREC 5, all scores are located in each of the three narrow value intervals: (10,100–10,000), (9010–9000), and (8010–8000); In harris1 of TREC 6, all scores are in a very narrow range of (99,999–97,000); and colm1 and colm4 in TREC 5 do not provide scoring information at all. In tnout9t2lc50 of TREC 9, the first one or several top-ranked documents may have scores as high as over 100,000, but all the rest just have scores of less than 2000. We have but only pointed out a few of the diverse types of abnormalities present. Therefore, we mainly use the binary logistic model to generate estimated scores for documents at different ranks.

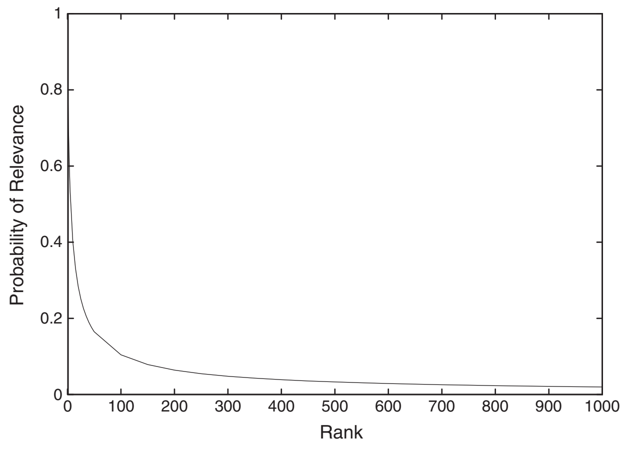


Fig. 1. Example of the function composition in Eq. (4) ($b<0$).

Table 1
Information of the five groups of runs submitted to TREC.

Group	Total number of runs	Number of selected runs	Number of queries
TREC 5 (ad hoc)	61	53	50
TREC 6 (ad hoc)	74	57	50
TREC 7 (ad hoc)	103	82	50
TREC 8 (ad hoc)	129	108	50
TREC 9 (web)	105	53	50

The binary logistic model works like this: for all the runs submitted in each year, we divide them into two groups, odd-numbered query group and even-numbered query group. All the runs in one group are put together as one super-run. Then we use $ln(t)$ as the independent variable and $score(t)$ as the dependent variable to calculate the coefficients of the logistic model. The coefficients we obtain are shown in Table 2.

Note that in any group, for any query and any result (run), the score assigned to any document is only decided by the ranking position of the document. For example, for the odd-numbered query group in TREC 7, we obtain $a=1.209$ and $b=-0.764$. It follows from Eq. (4) that

$$probability(t) = \frac{1}{1+e^{-1.209+0.764*ln(t)}} \tag{5}$$

Therefore, $probability(1)=0.7717$, $probability(2)=0.6655$,…, and so on. With such normalized scores (probabilities) for the runs submitted, the next step is to use multiple linear regression to train system weights and carry out fusion experiments. For all 50 queries in each year group, we divided them into two groups again: odd-numbered queries and even-numbered queries. First we used odd-numbered queries for training and even-numbered queries for testing data fusion methods; then we exchanged their positions by using even-numbered queries to train system weights and odd-numbered queries to test data fusion methods.

The file for multiple linear regression is a p row by $(n+1)$ column table. p is the total number of documents (duplicates are not counted) in all results and n is the total number of component systems involved in the fusion process. In the table, any element s_{ij} $(1 \le j \le n)$ represents the score that document d_i obtains from component system ir_j. Any element $s_{i(n+1)}$ is the judged score of d_i. The file can be generated by applying the following rules: 1. Documents are put into the table query by query. 2. For a given query, if document d_i occurs in at least one of the component results, then d_i is put into the table as a row. 3. If document d_i occurs in result r_j, then the score s_{ij} is calculated out by the logistic model (parameters are shown in Table 2). 4. If document d_i does not occur in result r_j, then a score of 0 is assigned to s_{ij}. 5. Ignore all those documents that are not retrieved by any component systems.

Apart from the linear combination method with the trained weights by multiple Regression (referred to as LCR), CombSum [9,10], CombMNZ [9,10], PosFuse [16], MAPFuse [16], SegFuse [23], the linear combination method with performance level weighting (referred to as LCP [3,24]), and the linear combination method with performance square weighting (referred to as LCP2, [29]) are also involved in the experiment. For LCP and LCP2, we also divide all queries into two groups: odd-numbered queries and even-numbered queries. A run's performance measured by mean average precision on odd-numbered queries ($MAP(odd)$) is used for the weighting of even-numbered group and vice versa. That is, for any combination we let

$$weight_{LCP}(odd) = MAP(even)$$

$$weight_{LCP}(even) = MAP(odd)$$

$$weight_{LCP2}(odd) = MAP(even) * MAP(even)$$

$$weight_{LCP2}(even) = MAP(odd) * MAP(odd)$$

Table 2
Coefficients obtained using binary logistic regression.

Group	Odd-numbered group		Even-numbered group	
	a	*b*	*a*	*b*
TREC 5	0.904	−0.670	0.762	−0.727
TREC 6	1.048	−0.722	0.594	−0.729
TREC 7	1.209	−0.764	1.406	−0.771
TREC 8	1.218	−0.796	1.538	−0.750
TREC 9	.448	−0.757	.368	−0.690

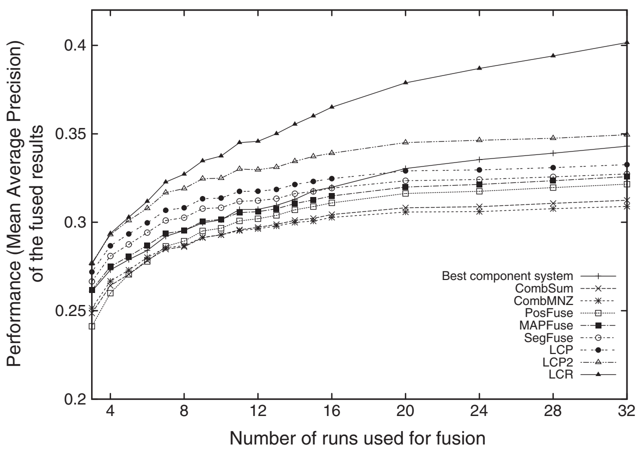


Fig. 2. Performance comparison of different data fusion methods (average of 5 groups, mean average precision, 200 randomly selected combinations).

Similarly, the same methodology is used in PosFuse, MAPFuse, and SegFuse.

We investigated the situation of 3–32 component systems. For each given number (3–16, 20, 24, 28, 32), 200 combinations were randomly selected and tested. Eight metrics, including average precision over all relevant documents, recall-level precision, precision at 5, 10, 15, 20, 30, and 100 document level, were used for performance evaluation.

Figs. 2–4 show MAP, RP, and P10 of all data fusion methods for all 5 year groups together. Results with other metrics are not presented due to space limitation and similarity to the figures presented.

From Fig. 2, we can see that LCR is the best in all year groups. If we consider the average of all the combinations (5 year groups*18 different numbers of component systems*200 combinations for each given number of component systems*50 queries), then LCR outperforms LCP2, LCP, SegFuse, MAPFuse, PosFuse, CombSum, CombMNZ, and the best component systems by 5.81%, 9.83%, 11.75%, 13.70%, 16.27%, 17.79%, 17.96%, and 12.35%, respectively. Two-tailed *T* test (see Table 3) shows that the difference between LCR and the others are very highly significant (p value < 0.01).

Considering the difficulty of beating the best component systems, the average improvement of 12.35% is a very good result. More importantly, such an improvement is consistent across all metrics and all year groups. On average, LCP and LCP2 also outperform the best component system by 2.33% and 6.25%, respectively. This confirms that LCP and especially LCP2 are also effective data fusion methods. However, they are not as consistent as LCR when different numbers of systems are fused or different metrics are used for evaluation. When 3 or 4 component systems are fused, LCP2 is almost as good as LCR. However, when more than 4 component systems are fused, LCP2 is not as good as LCR and the difference between LCP2 and LCR becomes greater and greater. SegFuse is marginally better than the best component system (by 0.56%), while the other four methods MAPFuse, PosFuse, CombSum, and CombMNZ are not as good as the best component system. See below for more discussion of them when recall-level precision and precision at 10 document level are used for retrieval evaluation.

When using the metric of recall-level precision (see Fig. 3), the improvement rates of LCR over other methods are: 5.03% (LCP2), 8.22% (LCP), 8.75% (the best system), 10.02% (SegFuse), 10.34% (MAPFuse), 12.80% (PosFuse), 14.40% (CombSum), and 14.51% (CombMNZ). On average, LCP and LCP2 perform slightly better than the best component system by 1.80% and 4.77%,

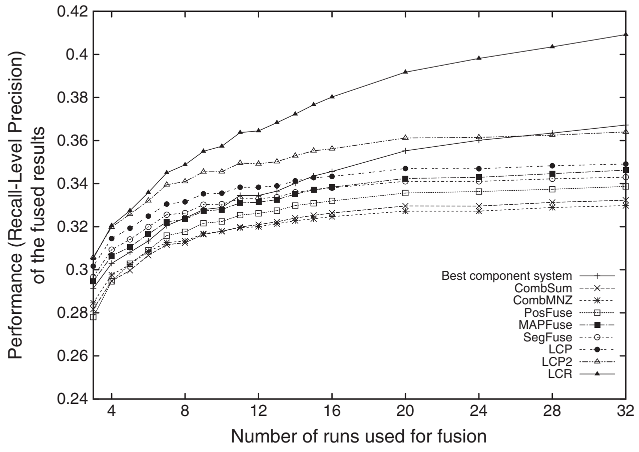


Fig. 3. Performance comparison of different data fusion methods (average of 5 groups, recall-level precision).

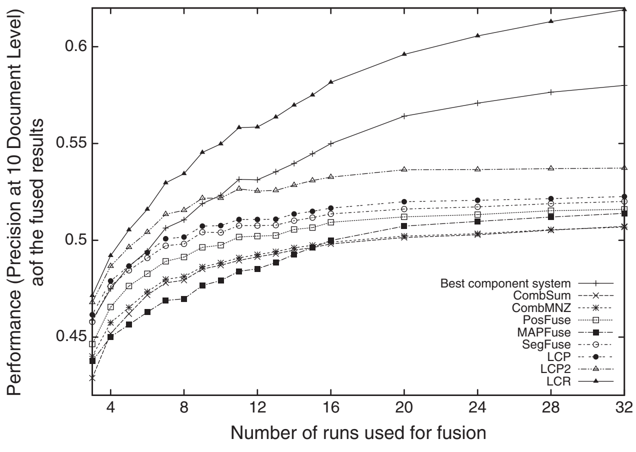


Fig. 4. Performance comparison of different data fusion methods (average of 5 groups, precision at 10 document level).

respectively. However, LCP is not as good as the best component system when 15 or more systems are fused, LCP2 is not as good as the best component system when 28 or 32 systems are fused. All other fusion methods are not as good as the best component system on average.

If considering precision at 10 document level (see Fig. 4), then the improvement rate of LCR over other methods are: 6.88% (LCP2), 9.76% (LCP), 5.10% (the best system), 10.38% (SegFuse), 14.85% (MAPFuse), 11.70% (PosFuse), 14.46% (CombSum), and 13.99% (CombMNZ). Apart from LCR, all other fusion methods are worse than the best component system on average, though LCP and LCP2 perform a little better than the component system when a few component systems are fused.

In order for us to have a better understanding of the properties of all the data fusion methods involved, we investigated the relation of data fusion methods to the best component system and to all component systems. We calculated Pearson product–moment correlation coefficients of performances between data fusion methods and the best component system, and coefficients between data fusion methods and the average of all component systems. A good data fusion method should have a strong positive correlation with the best component system rather than the average of all component systems. The result of this experiment is shown in Tables 4 and 5. Those coefficients vary from one year group to another even for the same data fusion method. However, in all year groups, LCR has the biggest Pearson's coefficients with the best component system, while it has the smallest Pearson's coefficients with the average of all component systems. This confirms that LCR is the best fusion method from a different angle.

By looking at the experimental results evaluated by different metrics (average precision, recall-level precision, precision at 10 document level, etc.,) at the same time, we have a few further observations.

1. We can see that LCR consistently outperforms all other methods and the best component system. When the number of component systems increases, the difference in performance between LCR and all other data fusion methods increases accordingly. When 32 systems are fused, LCR outperforms other data fusion methods by very large margins:

Pair of methods	MAP	RP	P10
LCR & LCP2	14.93%	12.45%	15.28%
LCR & LCP	20.78%	17.25%	18.44%
LCR & SegFuse	22.68%	19.28%	19.01%
LCR & MAPFuse	23.25%	18.15%	20.46%
LCR & PosFuse	24.92%	20.80%	19.95%
LCR & CombSum	28.58%	23.15%	22.06%
LCR & CombMNZ	29.97%	24.10%	22.15%
LCR & the best component system	17.04%	11.40%	6.71%

Table 3
Results of two-tailed *T* test of comparing LCR and three other data fusion methods (200 randomly selected combinations for 3–16, 20, 24, 28, and 32 component systems; &&degrees of freedom are 19,999; in all cases, *p* value<2.2e−16).

Pair of methods	*t* value	Mean of the differences
LCR & SegFuse	81.57	0.0363
LCR & LCP	71.86	0.0306
LCR & LCP2	54.70	0.0188
LCR & the best system	119.90	0.0377

Table 4
Pearson's correlation coefficient of performances (mean average precision) between data fusion methods and the best component system.

Methods	Average	Maximum	Minimum
CombSum	0.7376	0.7846	0.6870
CombMNZ	0.7164	0.7430	0.6568
PosFuse	0.7922	0.8553	0.7259
MAPFuse	0.8624	0.9225	0.8031
SegFuse	0.8353	0.8875	0.7616
LCP	0.8523	0.9091	0.7809
LCP2	0.8293	0.9639	0.8502
LCR	0.8947	0.9738	0.8468

2. LCP2 is always better than LCP, which confirms that performance square weighting is better than performance level weighting.
3. Compared with the best component system, LCP and LCP2 perform better when a small number of component systems are fused. However, there are significant differences when different metrics are used. Average precision is the metric that favours LCP and LCP2, while precision at 10 document level favours the best component system.
4. CombSum and CombMNZ perform badly compared with all the linear combination data fusion methods and the best component system. The result is understandable since no training is needed for both of them. It also demonstrates that treating all component systems equally is not a good fusion policy when there are a few poorly performing component systems.
5. As to effectiveness, the experimental results demonstrate that we can benefit from fusing a large group of component systems no matter which fusion method we use. Generally speaking, the more component systems we fuse, the better the fusion result we can expect. This can be seen from all those curves that are increasing with the number of component systems in all figures and tables.
6. It suggests that LCR can cope well with all sorts of component systems that are very different in effectiveness, even some of the component systems are very poor. It can bring effectiveness improvements steadily for different groups of component systems no matter which rank-based metric is used.

Finally, let us consider the situation in which some top results are involved in data fusion. In each year group, a few top runs are always the most noticeable since they are representatives of the state-of-the-art technology. One interesting question for data fusion is: if some top runs are involved, how it will affect the result of data fusion methods? Therefore, from all the 200 combinations in each setting, we choose those that include the best component result and observe the data fusion results of those sub-groups.

“ETHme1”, “CLAUG”, “CLARIT98COMB”, and “orc199man” are the best and their performances (MAP) are 0.3165, 0.3742, 0.3702, and 0.4130, in TRECs 5–8, respectively. In TREC 5, we obtain 10(3), 16(4), 23(5), 18(6), 29(7), 28(8), 34(9), 47(10), 45 (11), 44(12), 51(13), 50(14), 60(15), 41(16), 70(20), 97(24), 104(28), and 120(32) combinations. The figures in parentheses are the total number of component results that are involved in the data fusion experiment. For TREC 6, the figures are 14, 22, 15, 19, 23, 34, 33, 48, 33, 37, 34, 52, 57, 56, 87, 80, 90, 113; For TREC 7, they are 5, 12, 8, 8, 14, 21, 25, 24, 28, 32, 17, 40, 44, 55, 60, 59, 75, 72; and for TREC 8, they are 10, 10, 14, 15, 10, 13, 13, 19, 19, 16, 25, 19, 21, 28, 39, 46, 49, 72. Fig. 5 presents the result of the LCR data fusion method.

From Fig. 5, we can see that LCR steadily outperform the best component result in all four year groups. When 3 component results are fused, the improvement rates of LCR over the best component result in each year group are 5.50%, 3.04%, 4.92%, and 7.51%, respectively; when 32 component results are fused, the improvement rates are 12.07% (TREC 5), 35.20% (TREC 6), 19.41% (TREC 7), and 21.75% (TREC 8). Compared with the original group (a total of 200 combinations), there is some difference. Originally, the improvement rates for 3 component results are 6.75% (TREC 5), 6.73% (20.44%), 6.30% (TREC 7), and 5.25% (TREC 8); the improvement rates for 32 component results are 18.01% (TREC 5), 20.44% (TREC 6), 23.06% (TREC7), and 19.03% (TREC 8). It suggests that LCR is still very effective when top component results are involved for data fusion.

Table 5
Pearson's correlation coefficient of performances (mean average precision) between data fusion methods and the average of all component systems.

Methods	Average	Maximum	Minimum
CombSum	0.6782	0.7233	0.6172
CombMNZ	0.6593	0.7076	0.5770
PosFuse	0.4763	0.5513	0.4154
MAPFuse	0.5627	0.6652	0.4949
SegFuse	0.5398	0.6659	0.4463
LCP	0.5194	0.6573	0.4335
LCP2	0.4936	0.5962	0.3959
LCR	0.4114	0.5244	0.3261

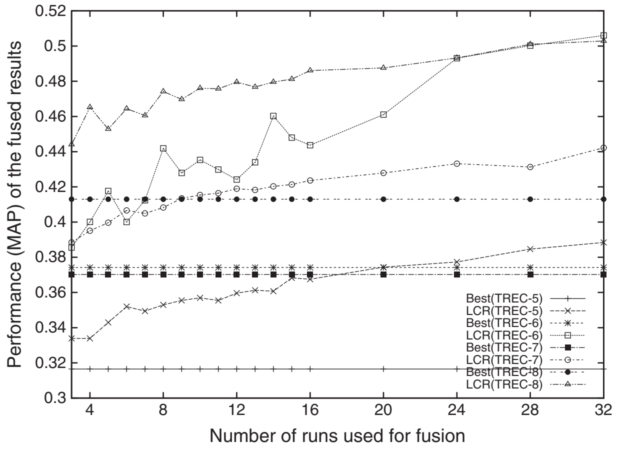


Fig. 5. Performance of LCR when the best component result is involved in data fusion (TRECs 5–8).

5. Score normalization methods

Score normalization methods definitely have impact on the performance of the data fused methods. In Section 4, we have demonstrated that using multiple regression to obtain system weights is an effective technique for the linear combination data fusion method to improve performance, with the help of using logistic regression to generate relevance scores. In this section, we investigate and evaluate some more methods of normalizing raw scores and generating scores from ranking information. One way of evaluating those methods is: for every document involved, we compare its estimated score (obtained from a given normalization method) with the judged score. The Euclidean distance can be used for this purpose. For all the runs selected in each year group, we calculate the Euclidean distance of every resultant list for a given query. That is,

$$dist = \sqrt{\sum_{i=1}^{1000} (e\_score(d_i) - j\_score(d_i))^2}$$

Apart from the binary logistic model, we also include the cubic model, 0–1 linear normalization, 0.02–0.6 linear score normalization, 0–1 Borda, and 0.02–0.6 Borda. We shall explain them one by one. The cubic model uses the equation

$$score(t) = a_0 + a_1 ln(t) + a_2 ln(t)^2 + a_3 ln(t)^3$$

to estimate the relation between ranks and relevance scores. Here t is the rank position of a document, and $score(t)$ is the estimated score of documents at rank t. For the cubic model, the methodology for obtaining those parameters is just the same as the logistic model. Some other models such as linear, logarithmic, inverse, and exponential, are also tried, but they are not as good as the binary logistic model and the cubic model. 0–1 linear normalization method uses Eq. (1) to calculate scores for all documents involved. In fact, this method can be improved for the TREC scenario since top-ranked documents are not always relevant while bottom-ranked documents are not always irrelevant. Instead of using 1 and 0 for the upper and lower limits, we use a and b as upper and lower limits, where $(1>a>b>0)$. In the experiment, 0.6 and 0.02 are chosen arbitrarily as upper and lower limits. For Borda count, its scores range from 1000 to 1, and Euclidean distance cannot be calculated directly. Therefore, we use either 0–1 linear normalization or 0.02–0.6 linear normalization to further normalize them. They are referred to as 0–1 Borda and 0.02–0.6 Borda. Table 6 shows the average Euclidean distances of all resultant lists for all the runs involved in all four year groups.

From Table 6, we can see that 0–1 Borda is the worst with a distance of between 17 and 18 in all 4 year groups. 0.02–0.6 Borda looks better than 0–1 Borda with a difference of between 11 and 12 in all year groups. 0.02–0.6 linear normalization is better than 0–1 linear normalization by 1 to 1.5 units in Euclidean distance. The logistic model and the cubic model are the best, and distances

Table 6
Euclidean distance of estimated scores and judged scores with different score normalization methods for four groups of runs in TREC.

Method	TREC 5	6	7	8
0–1 Linear	8.31	7.57	7.53	7.55
0.02–0.6 Linear	6.84	6.33	6.53	6.40
0-1 Borda	17.64	17.72	17.59	17.50
0.02–0.6 Borda	11.37	11.34	11.42	11.37
Cubic	5.80	5.43	5.96	6.37
Logit	5.82	5.42	5.95	6.38

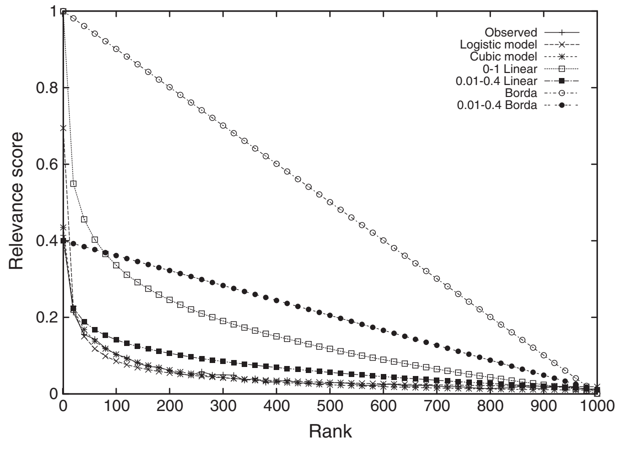


Fig. 6. Observed curves of relevance scores vs. estimated curves in TREC 5.

in both cases are very close. However, it should be noted that the Euclidean distance is not 100% reliable for determining the goodness of any score normalization method. For all the scores involved, if we multiply all of them by a fixed number, then the fused results will not be changed since only the ranking of those documents matters. Therefore, 0–1 linear normalization and 0–0.6 linear normalization should be equally good, and the difference between 0–1 linear normalization and 0.02–0.6 linear normalization should be very small though the difference between them may look greater by the Euclidean distance. It is the same for 0–1 Borda and 0.02–0.6 Borda.

In order to have a closer look at the relation between relevance scores and ranks, we present the observed score curve and estimated curves for the TREC 5 data collectively, in Fig. 6. The observations for the three other year groups is very similar and therefore not presented.

In Fig. 6, the observed curve is the one with many twists. From Fig. 6, we can see that 0–1 Borda is not very good since all scores are over-estimated considerably. Moreover, the rates of over-estimation vary at different ranking positions. It can be improved a little by using better upper and lower limits such as 0.02 and 0.6. In fact, according to the observation, 0.01 and 0.4 would be approximately the best upper and lower limits for TREC 5 data. The Euclidean distance for 0.01–0.4 Borda is 8.52, much shorter than 17.64 (0–1 Borda) and 11.37 (0.02–0.6 Borda). However, there is no way for a variation of Borda to approach other methods. 0–1 linear score normalization looks much better than 0–1 Borda. If we set proper upper and lower limits for the linear score normalization method, we can obtain even better results. For TREC 5 data, the distance for 0.02–0.6 Linear is 6.84, and the distance for 0.01–0.4 Linear, whose curve is just a little over the observed one, is 6.19. However, even 0.01–0.4 Linear, which is almost the best for linear score normalization methods, is not as good as the logistic regression model and the cubic regression model. We hypothesize that it is mainly caused by the abnormal score distributions in some of the runs submitted. Finally, the logistic model and the cubic model are the two best. Both of them fit the observed curve well.

We carried out some data fusion experiments to compare those score normalization (generating) methods. Firstly, we tested CombSum by using different score normalization (generating) methods. As before, for each given number (3–15), 200 combinations were randomly selected and tested. Fig. 7 shows the results with TREC 6 data. It can be seen that Borda normalization

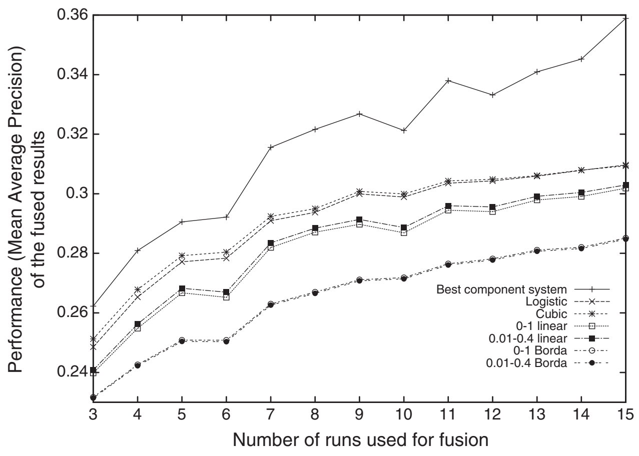


Fig. 7. Fusion performance (MAP) of CombSum in TREC 6 with several different score normalization methods.

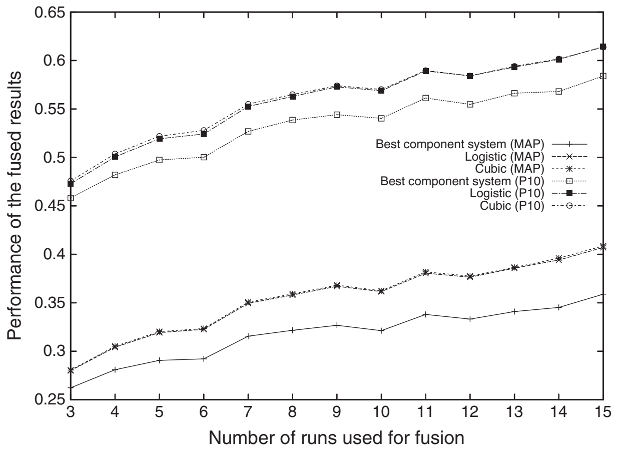


Fig. 8. Comparison of fusion performance of the linear combination method in TREC 6 using logistic model and cubic model to generate scores.

methods (0–1 Borda and 0.01–0.4 Borda) are the worst, linear score normalizations (0–1 Linear and 0.01–0.4 Linear) are in the middle, and the cubic model and the logistic model are the best two. The difference between the cubic/logistic model and Borda is about 10%, and the difference between the cubic/logistic model and the linear score normalization model is about 3%. Similar differences are also observed when other metrics are used.

Secondly, we used both logistic model and cubic model for the linear combination data fusion method. For both models, the weights for all component systems are trained by using 25 queries and tested by using the other 25 queries. The methodologies are just the same as before. Fig. 8 shows the average performance of 200 randomly selected runs for each of the given number of component systems with the TREC 6 data set. From Fig. 8, we can see the cubic model and the logistic model are equally good by both metrics MAP and P10.

Overall, from the experimental results presented in this section, we demonstrate that: if relevance scores can be normalized or generated in one way or another, then multiple linear regression is able to use them to obtain suitable weights for the linear combination data fusion method.

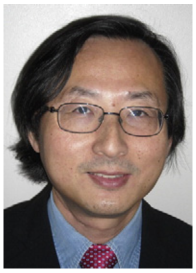
6. Conclusions

In this paper we have presented a new fusion method that uses the multiple linear regression to obtain suitable weights for the linear combination method. The weights obtained in this way is optimum in the least squares sense of estimating relevance scores of all related documents by linear combination. The extensive experiments with five groups of TREC runs demonstrate that it is a very good data fusion method. It outperforms other major data fusion methods such as CombSum, CombMNZ, PosFuse, MAPFuse, SegFuse, the linear combination method with performance level weighting or performance square weighting, and the best component system, by large margins. We have also investigated methods of generating scores from ranking information and raw score normalization methods, and a comparison has been made between a group of such methods. We find that the binary logistic model and the cubic model are better than the 0–1 linear score normalization method for the data set involved. The latter has been commonly used in a variety of data fusion investigations and experiments. We believe this piece of work, especially the part of using multiple linear regression to find appropriate weights for linear combination, is a significant refinement of the data fusion technique in information retrieval and has laid a solid foundation for the development of more effective information retrieval systems.

References

[1] A. Arampatzis, J. Kamps, A signal-to-noise approach to score normalization, Proceedings of the 18th Annual International ACM CIKM Conference, Hong Kong, China, 2009, pp. 797–806.
[2] J.A. Aslam, M. Montague, Models for metasearch, Proceedings of the 24th Annual International ACM SIGIR Conference, New Orleans, Louisiana, USA, 2001, pp. 276–284.
[3] B.T. Bartell, G.W. Cottrell, R.K. Belew, Automatic combination of multiple ranked retrieval systems, Proceedings of ACM SIGIR'94, Dublin, Ireland, 1994, pp. 173–184.
[4] A.L. Calvé, J. Savoy, Database merging strategy based on logistic regression, Information Processing and Management 36 (2000) 341–359.
[5] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, Proceedings of the Tenth International World Wide Web Conference, Hong Kong, China, 2001, pp. 613–622.
[6] M. Efron, Generative model-based metasearch for data fusion in information retrieval, Proceedings of the 2009 Joint International Conference on Digital Libraries, Austin, USA, 2009, pp. 153–162.
[7] G.J. Fakas, A novel keyword search paradigm in relational databases: object summaries, Data & Knowledge Engineering 70 (2011) 208–229.
[8] M. Farah, D. Vanderpooten, An outranking approach for rank aggregation in information retrieval, Proceedings of the 30th ACM SIGIR Conference, Amsterdam, The Netherlands, 2007, pp. 591–598.

[9] E.A. Fox, M.P. Koushik, J. Shaw, R. Modlin, D. Rao, Combining evidence from multiple searches, The First Text REtrieval Conference (TREC-1), Gaitherburg, MD, USA, 1993, pp. 319–328.
[10] E.A. Fox, J. Shaw, Combination of multiple searches, The Second Text REtrieval Conference (TREC-2), Gaitherburg, MD, USA, 1994, pp. 243–252.
[11] G. Solskinnsbakk, a.J.A.G., Combining ontological profiles with context in information retrieval, Data & Knowledge Engineering 69 (2010) 251–260.
[12] J.L. Hong, E. Siew, S. Egerton, Information extraction for search engines using fast heuristic techniques, Data & Knowledge Engineering 69 (2010) 169–196.
[13] Z. Ji, Q.M. Wu, An improved artificial immune algorithm with application to multiple sensor systems, Information Fusion 11 (2010) 174–182.
[14] A. Juarez-Gonzalez, M.M. y Gomez, L. Pineda, D. Avendano, M. Perez-Countino, Selecting the n-top retrieval result lists for an effective data fusion, Proceedings of 11th International Conference on Computational Linguistics and Intelligent Text Processing, Iasi, Romania, 2010, pp. 580–589.
[15] J.H. Lee, Analysis of multiple evidence combination, Proceedings of the 20th Annual International ACM SIGIR Conference, Philadelphia, Pennsylvania, USA, 1997, pp. 267–275.
[16] D. Lillis, L. Zhang, F. Toolan, R. Collier, D. Leonard, J. Dunnion, Estimating probabilities for effective data fusion, Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Geneva, Switzerland, 2010, pp. 347–354.
[17] R. Manmatha, T. Rath, F. Feng, Modelling score distributions for combining the outputs of search engines, Proceedings of the 24th Annual International ACM SIGIR Conference, New Orleans, USA, 2001, pp. 267–275.
[18] M. Montague, J.A. Aslam, Relevance score normalization for metasearch, Proceedings of ACM CIKM Conference, Berkeley, USA, 2001, pp. 427–433.
[19] M. Montague, J.A. Aslam, Condorcet fusion for improved retrieval, Proceedings of ACM CIKM Conference, McLean, VA, USA, 2002, pp. 538–548.
[20] H. Nottelmann, N. Fuhr, From retrieval status values to probabilities of relevance for advanced ir applications, Information Retrieval 6 (2003) 363–388.
[21] M.E. Renda, U. Straccia, Web metasearch: rank vs. score based rank aggregation methods, Proceedings of ACM 2003 Symposium of Applied Computing, Melbourne, USA, 2003, pp. 841–846.
[22] A.K. Seewald, How to make stacking better and faster while also taking care of an unknown weakness, Proceedings of the 19th International Conference on Machine Learning, Sydney, Australia, 2002, pp. 554–561.
[23] M. Shokouhi, Segmentation of search engine results for effective data-fusion, Advances in Information Retrieval, Proceedings of the 29th European Conference on IR Research, Rome, Italy, 2007, pp. 185–197.
[24] P. Thompson, Description of the PRC CEO algorithms for TREC, The First Text REtrieval Conference (TREC-1), Gaitherburg, MD, USA, 1993, pp. 337–342.
[25] K.M. Ting, I.H. Witten, Issues in stacked generalization, Artificial Intelligence Research 10 (1999) 271–289.
[26] C.C. Vogt, G.W. Cottrell, Predicting the performance of linearly combined IR systems, Proceedings of the 21st Annual ACM SIGIR Conference, Melbourne, Australia, 1998, pp. 190–196.
[27] C.C. Vogt, G.W. Cottrell, Fusion via a linear combination of scores, Information Retrieval 1 (1999) 151–173.
[28] S. Wu, A geometric probabilistic framework for data fusion in information retrieval, Proceedings of the 10th International Conference on Information Fusion, Quebec, Canada, 2007, pp. 1–8.
[29] S. Wu, Y. Bi, X. Zeng, L. Han, Assigning appropriate weights for the linear combination data fusion method in information retrieval, Information Processing and Management 45 (2009) 413–426.
[30] S. Wu, S. McClean, Improving high accuracy retrieval by eliminating the uneven correlation effect in data fusion, Journal of the American Society for Information Science and Technology 57 (2006) 1962–1973.
[31] S. Yang, M. Wang, L. Jiao, Z. Wang, Image fusion based on a new contourlet packet, Information Fusion 11 (2010) 78–84.
[32] H. Zhao, S. Ram, Combining schema and instance information for integrating heterogeneous data sources, Data & Knowledge Engineering 61 (2007) 281–303.
[33] D. Zhou, S. Lawless, J. Min, V. Wade, A late fusion approach to cross-lingual document re-ranking, Proceedings of the 19th ACM Conference on Information and Knowledge Management, Toronto, Canada, 2010, pp. 1433–1436.



Shengli Wu is a lecturer at the University of Ulster at Jordanstown, the United Kingdom. He obtained his Ph.D. degree from the Department of Computer Science and Engineering, Southeast University, China in 1996. His research areas include database and information systems, information retrieval, and data mining.