Standing on the Shoulders of Giants
Ranking by Combining Multiple Sources*

Yaoshuang Wang Xiaoguang Qi Brian D. Davison
Department of Computer Science & Engineering
Lehigh University
{yaw206,xiq204,davison}@cse.lehigh.edu

June 2007

Abstract

In most ranking systems for information retrieval, there are multiple signals that need to be combined to generate a single score for ranking a result. In this work we consider how the output score of various ranking systems can be combined to rank the TREC 2003 and 2004 benchmark datasets. Not surprisingly, we find that the performance of a combined system is able to achieve significantly better performance than any single ranker. We also find that the best performing combination is that of a form of weighted sum of a link-based and text-based method, in which the text-based method is given most of the weight.

In addition to combining the scores of multiple rankers directly, we also consider their use as features fed to a classifier. Interestingly, while classifiers perform classification poorly on the benchmark data, they can be used to benefit the ranking problem. We find that the scoring generated by a linear SVM classifier can perform respectably for ranking, and when used in combination with another ranking feature, can outperform all other tested ranking combinations.

1 Introduction

In information retrieval, a ranking algorithm is a function that, given a query, returns a list of documents ordered by relevance to the query. Due to the abundance of searchable information on the Web, a query may have millions of matching documents. Therefore, besides finding out which documents are relevant, it is also important to rank highly the important documents. As a result, ranking plays a crucial role in search engines. PageRank [13], for example, was central in Google's rise as a leading search engine.

*Technical Report LU-CSE-07-011, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015.

Dozens, if not hundreds, of ranking algorithms have been proposed in previous research. Although some of them have been shown to be quite effective, Lebanon and Lafferty [8], Bartell et al. [2], Rapela [16] and others have demonstrated that combining multiple ranking sources may achieve even better performance. The intuition is old and simple: "two heads are better than one". Thus, decisions backed by a pool of experts are more likely to be sound than those made by any individual expert. This idea has already been practiced by commercial engines. For example, Google claims hundreds of factors in calculating the results of a query [3, 4]. In web search, combining multiple ranking algorithms is useful for at least two reasons:

- each ranking algorithm can take a different view of the data; therefore combining them may provide a more collective view; and
- each ranking algorithm may be good at answering some types of queries; therefore combining them could provide better answers to more queries.

The important questions are what to combine and how to combine. In this study, we are mainly interested in the combination of two types of rankings: rankings based on information retrieval approaches (text-based rankings) and rankings based on hyperlink analysis (link-based rankings). In our intuition, text-based rankings and link-based rankings focus on different aspects of web data (content and hyperlink structure, respectively). As a result, text-based rankings tend to promote the textually relevant documents, while link-based rankings tend to promote the important documents. Therefore, although the combination of two text-based rankings (or two link-based rankings) may get good results, we argue that combination across these two groups has a better potential. The combination of text-based rankings and link-based rankings has been a common practice in web information retrieval. However, as far as we know, relatively few papers performed extensive study to support such a combination (e.g., [10]).

As to the remaining question, how to combine, we propose two methods: arithmetic combination and classification. The first method adjusts the weight of scores generated by different ranking functions and combines them using arithmetic functions to get the combined score. The second transforms the ranking problem into a categorization problem, and utilizes off-the-shelf data mining approaches to combine the rankings. Our experiments showed that both methods make significant improvements (12% and 6%, respectively, in terms of NDCG@10) over the baseline (best ranking based on any single algorithm). We continued to combine the two methods and achieved even better performance (19% in NDCG@10 over the baseline).

The contributions of our paper are:

- a rank fusion algorithm that combines arithmetic functions and machine learning approaches, which outperforms many exsiting ranking algorithms on the TREC datasets;
- an empirical study of the performance of a number of arithmetic combining functions in web ranking;

- an approach to rank fusion via supervised learning.

The rest of this paper is organized as follows: related work is introduced in Section 2; the dataset used in our experiments is described in Section 3; the arithmetic combination approach and classification approach are presented in Sections 4 and 5, respectively; in Section 6, we present the combination of the two approaches; finally, we conclude with a discussion in Section 7.

2 Related Work

In a search engine ranking system, some features play a more critical role than others. Search engine ranking systems have different concerns from traditional information retrieval ranking systems. Besides query relevance, which is reflected in the text-based feature scores, search engine ranking systems care about documents' authority, and so must combine the two measures to generate a final ranking. Many rank fusion methods have been proposed by prior work. These previous works have one or more of the following characteristics: ranking by feature scores or transformed feature scores, ranking by feature rank score, ranking by training classification results [22]. Most previous works focused on improving information retrieval precision, recall or other performance metrics through various ways of using specific features.

The linear combination of unweighted min, max and median by Fox and Shaw [20] and the CombSUM and CombMNZ by Lee [9] are based on the assumption that results from different search engines have certain similarity. But in the TREC datasets, this assumption doesn't always hold. Among the text-based features, there is similarity from query term frequency. But between some text-based features and link-based features, the existence of similarity may not be true. Renda and Stracia [17] used the same Min-Max based feature score and rank normalization as we did in our work. However, we extended the rank-based methods in some ways including rank discretization and reciprocal rank. In addition, we studied several different variations based on normalized features and normalized rank, which resulted in a more extensive exploration of the feature score space. A number of researchers have used the TREC 2003 and 2004 topic distillation datasets [21, 15, 14, 24]. We will compare with their results in section 7.

3 Datasets utilized

In many years, the Text REtrieval Conference (TREC) has included a Web track. In 2003 and 2004, that track utilized a 2002 snapshot of about 1M .GOV web pages. LETOR [11] is a package of benchmark datasets for LEarning TO Rank. We used the topic distillation queries for TREC 2003 and 2004 , TD-2003 and TD-2004. The LETOR data includes 44 precalculated feature scores (some of which we select for use as described in Table 1), which we use in both the weighted feature score combination approach and the classification approach.

4 Ranking fusion method

4.1 Feature selection

We filtered the TREC features and distilled the most interesting and critical features based on their performance and representativeness of different types. We built a subset of 14 features out of the 44 features in the TREC dataset. The subset includes the most important text-based features and link-based features. In Table 1 BM25, NEW BM2501, SCORE2501, tfidf body and LMIR.DIR of anchor are mostly textual relevant features; the rest are more likely to be link-based features. We studied different weighted combinations of the two groups of features, found the most significant combinations and the preferred weight of each feature.

Feature[1] number	**Original[2] feature name**	**Avg MAP on 2003**	**Avg MAP on 2004**
1	BM25 [18]	0.1255	0.2818
6	HITS Authority [6]	0.0922	0.1002
7	HITS Hub [6]	0.0720	0.0576
13	NEWBM2501-0.8 [15]	0.1738	0.2898
14	PageRank[13]	0.0716	0.0661
17	LMIR.DIR of anchor [7]	0.1382	0.2196
27	SCORE2501-0.8 [15]	0.1725	0.2902
32	tfidf body [1]	0.1274	0.1490
36	Topical PageRank [12]	0.0849	0.0691
37	Topical HITS authority [12]	0.1073	0.1046
38	Topical HITS hub [12]	0.0381	0.0519
40	UIUCFEAPRO-1-0.9 [19]	0.1393	0.2781
42	UIUCFEAPRO-10-0.9 [19]	0.1596	0.2889
43	UIUCFEAPRO-11-0.9 [19]	0.1415	0.2622

Table 1: Selected features in our experiment

[1]The same feature numbers are also used in Tables 2 and 3.
[2]The original feature names or numbers in the TREC dataset.

4.2 Combining and normalizing features

It is more interesting for us to see how the combination of text-based score and link-based score can improve the ranking when compared to combinations of two scores from the same category. The individual feature score range varies significantly between different features. Some feature scores can be large and eventually dominate the final rank result no matter which other feature is used in the combination. Score normalization becomes necessary to smooth the score distribution for better score combination.

We looked at different ways of combining the transformed features in this section in order to find the best combination.

Notation

- f_i,f_j: *scores using feature from Table 1*
- R_i: *rank by feature i, in which the document with highest feature value* f_i *gets a lowest* R_i

1. **Log score transformation:**
   We explored two different combinations of log transformation.
   - Weighted sum of logarithm of feature scores:
     $logsum = a * log(1 + f_i) + (1 - a) * log(1 + f_j)$
   - Product of logarithm of feature scores:
     $logprod = log(1 + f_i) * log(1 + f_j)$
2. **Weighted sum of min-max score normalization**
   Due to the variation in score range of different features, performing score normalization may improve the rank results. We explored two different formats of score normalization. The first one, $normF_1$, is normalized based on the current feature's minimum and maximum score. This will normalize the original score to be within the range from 0 to 1. The $normF_2(i)$ does further normalization based on the $normF_1(i)$. The $normF_2(i)$ further decreases the distances between two scores.
   - $normF_1(i) = \frac{f_i - MIN(f_i)}{MAX(f_i) - MIN(f_i)}$
   - $normF_2(i) = \frac{normF_1(i)}{1 + normF_1(i)}$
3. **Product of Min-max normalized feature scores:** $normF_1P$
   Explore the product rank of the $normF_1$ from feature 1 to 14. We find $\max_{n=2}^{14}(\prod_{i=1}^{n} normF_1(i))$;
4. **Weighted sum rank score normalization**
   We performed two formats of rank score normalization. R_i has a score range from 1 to the total number of documents. The *normRank* normalizes R_i based on maximum rank. In both *normRank* and *reciprocalRank*, the documents with high ranks will get a higher score than documents with low ranks. The *normRank* has an even distance between two neighboring documents, but the *reciprocalRank* has a bigger distance between high ranking documents. We expect the different distance may generate final ranking results differently.

- $normRank(i) = \frac{MAXRANK - R_i}{MAXRANK}$
- $reciprocalRank(i) = \frac{1}{R_i}$

5. **Weighted sum log normalization of normalized rank score**

   - $lognormRank(i) = log(1 + \frac{MAXRANK - R_i}{MAXRANK})$

6. **Discretization of normalized feature score:** $normF_1Dis(i)$
   We used 101 fixed data points to represent feature values, such as 0, $\frac{1}{100}$, $\frac{2}{100}$, which are evenly distributed within the range of [0,1]. Feature values are rounded to the nearest fixed data point.
7. **Discretization of normalized rank scores:** $normRankDis(i)$
   Discretize the $normRank(i)$, which is also within the range of [0,1], to 100 discrete values within the same range.
8. **Weighted sum of** $normRank(i)$ **and** $normF_1(i)$**:** $normRankF_1Comb(i)$
9. **Combination of discretized** $normRank(i)$ **and discretized** $normF_1(i)$**:** $normRankF_1DisComb(i)$
10. **Rank by feature values:** $voteRank$
    Let all features vote for the documents to rank. Voting using just top-k $normRank$. In our study we used k=50, as we care about the top ranks most. The final decision is made up the sum of all votes.
11. **weighted exponential sums** $expSum(i,j)$

    - $expSum(i,j) = w_1 * e^{(1+normF_1(i))} + w_2 * e^{(1+normF_1(j))}$

12. **exponential combination** $expComb(i,j)$

    - $expComb(i,j) = normF_1(i)^{(w_1+normF_1(j))}$

13. **Rank by discretized and normalized feature values (**$normF_1Dis(i)$**):** $voteDisRank$
    The difference between $voteDisRank$ and $voteRank$ is that the rank decision is the result of the sum of discretized $normRank$.

4.3 Experiments in partial feature space

Based on TREC2003 and TREC2004 feature scores, we experimented with different methods and measured their ranking quality by MAP and NDCG@10. Table 1 contains the selected feature number and names. The Table 2 and Table 3 contains the performance metrics of different ranking methods. We experimented with various weighted combinations of all pairs of features. The values in both the column of average MAP and average NDCG@10 are the maximum values we found for that ranking algorithm. The feature combinations with the maximum average MAP are denoted by a superscript m. The feature combinations with the maximum average NDCG@10 are denoted by a superscript n.

[3] (i,j) means the combination of feature i and feature j. (i-j) means from feature i to feature j.

Method	Average MAP	Average NDCG@10	Best[3] pairwise combinations
logsum	0.1986	0.2673	$(2,4)^n$(3,4)(2,6) (3,6)$(4,9)^m$(6,9)
logprod	0.2085	0.2887	(2,7)(3,7)$(7,9)^{nm}$
$normF_1$	0.2019	0.2679	(3,6)(3,4)$(6,9)^n$ $(4,9)^m$(3,14)
$normF_2$	0.2086	0.2718	(6,9)(4,9)$(3,14)^m$
$normF_1P$	0.1886	0.2745	n=7
normRank	0.2100	0.2955	(6,9)(4,9)$(7,9)^{mn}$
reciprocalRank	0.1928	0.2554	(4,7)$(4,9)^n$ $(4,14)^m$(6,14)
lognormRank	0.2206	0.2985	$(1,8)^n$(1,10) (1,7)(1,11)$(1,9)^m$
$normF_1Dis$	0.2024	0.2698	(3,4)$(4,9)^{mn}$(6,9)
normRankDis	0.2250	0.3064	$(7,9)^{mn}$
normRank F_1Comb	0.2126	0.2805	(6,9)$(4,9)^{mn}$
$normRankF_1$ *DisComb*	0.2131	0.2858	(6,9)$(4,9)^{mn}$
voteRank	0.1906	0.2724	
voteDisRank	0.1739	0.2528	

Table 2: Performance of ranking combination methods on TREC2003 queries.

The remaining pairs of features are those that achieved at least 95% of the best performance.

Based on the values in Table 2 and Table 3, we consider the algorithm with the largest sum of MAP and NDCG@10 on both TREC2003 and TREC2004 as the best ranking algorithm.

- The overall best ranking algorithm is *normRank*; subsequent transformations based on *normRank* also performed well.
- The best feature pairs are (feature 27: SCORE2501-0.8, feature 37: Topical HITS authority) and (feature 13: NEWBM2501-0.8, feature 37: Topical HITS authority).
- The preferred weight of Topical HITS authority is around 6%.

4.4 Relationship between text-based and link-based features

Besides finding the best ranking fusion algorithm, we wanted to examine the patterns of combination between text-based features and link-based features. We experimented on the different combinations of typical text-based features

Method	Average MAP	Average NDCG@10	Best pairwise combinations
logsum	0.3085	0.39644	(1,12)(1,4)^m (4,12)^n(4,13) (6,13)(6,12)(12,13)
logprod	0.3072	0.3893	(1,4)^m(1,6)(1,12) (4,11)(4,13)(4,12)^n (6,11)(6,12)(6,13)
$normF_1$	0.3196	0.3965	(1,6)(1,4)(4,12)^n (4,14)(6,12)(6,14)^m
$normF_2$	0.3157	0.3964	(6,12)(6,11)(4,12)^n (1,12)(11,12)(1,4) (4,14)(6,14)^m
$normF_1P$	0.2818	0.33165	n=1
normRank	0.3179	0.4070	(11,12)(6,12)(4,12)^{mn} (4,9)(6,9)
reciprocalRank	0.3122	0.3837	(6,11)(1,11)(12,14) (4,11)(6,14)(4,14) (1,12)^m(4,12)^n (6,12)(11,14)(11,12)
lognormRank	0.3092	0.3866	(1,13)(2,3)^{mn}(1,14)
$normF_1Dis$	0.3176	0.3978	(6,12)(1,4)(4,12)^n (1,12)(4,14)(11,12) (6,14)^m
normRankDis	0.3085	0.3867	(6,14)(4,14)(1,12)^{mn}
normRank F_1Comb	0.3160	0.3996	(6,12)(4,12)^n (4,14)(6,14)^m (11,12)(4,9)(6,9)
$normRankF_1$ *DisComb*	0.3142	0.3990	(4,13)(6,11)(11,12) (4,14)(6,12)(4,12)^n (1,12)^m(6,14)
voteRank	0.2186	0.2858	
voteDisRank	0.2149	0.2826	

Table 3: Performance of ranking combination methods on TREC2004 queries.

and link-based features. In Figures 1, 2 and 3, the X axis shows the combinational weight of the text-based feature. The Y axis shows the NDCG@10. As shown in Figure 1, ranking performance can be improved by adding certain link-based features to text-based features. From Figures 1 and 2, we can see that New BM2501 combined with Topical HITS authority outperforms the combinations of most other link-based features in our experiment. Note that the overall best combination of features is not shown in Figures 1 and 2 as it was not among the best feature combinations in TREC2003. Figure 3 shows the topical HITS authority combined with different text-based features; the combinations with NEW BM2501 or SCORE2501 outperform the rest.

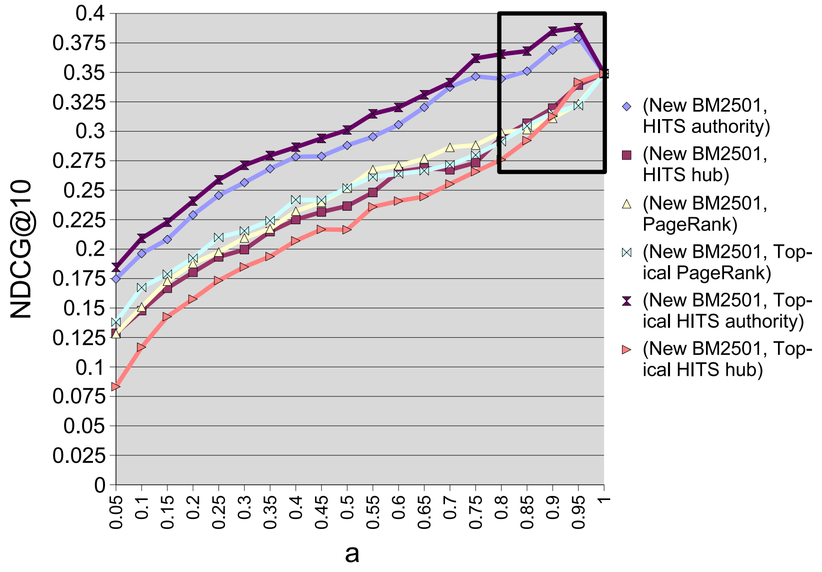

Figure 1: *normRank* of different feature combinations on TREC2004. (Marked inset amplified below in Figure 2).

4.5 Time complexity

Response time is crucial for a search engine ranking system. The ideal ranking fusion algorithm should be simple and effective at improving the ranking result. Even if some computationally intensive rank fusion methods could generate good results, it would not be practical for a search engine ranking system to apply them. The best ranking algorithm in our experiment is the MIN-MAX normalized rank score *normRank*, which consistently performed well for both TREC2003 and TREC2004 datasets. The most computationally expensive operation of the *normRank* algorithm is the feature scores sorting. Given a list of n documents, the time complexity of *normRank* is $\Theta(n\lg n)$. The actual cost of generating the feature value will likely be higher, so in comparison, *normRank*'s additional cost will be negligible.

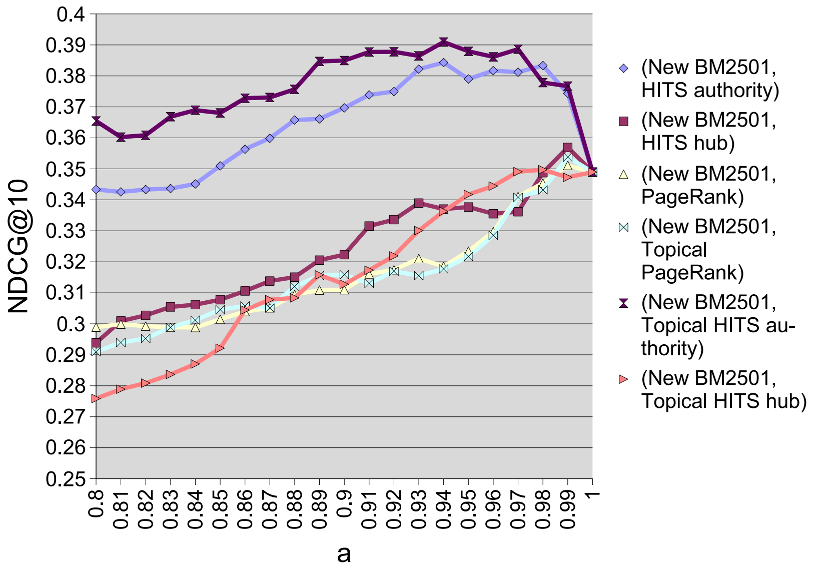

Figure 2: Amplified *normRank* of different feature combinations on TREC2004.

5 Ranking via classification

In the previous section, we showed that arithmetic combination is able to significantly boost the ranking performance. This approach is useful in that it reveals the relative weight of different scores. However, arithmetic combination might be too naive to model the relationships among the different features in the ranking process. In this section, we transform the ranking problem into classification problem, the latter of which can be solved by a variety of existing machine learning approaches.

By dividing the documents in the collection into two categories, relevant and irrelevant with respect to a given query, the ranking problem can be reduced to a binary classification problem, in which relevant documents are positive instances, irrelevant documents are negative instances, and scores of different ranking functions are features. Although this transformation may seem natural, there are potential problems.

- Categorization algorithms often optimize the model based on the observed class distribution (prior probability). Therefore, if there is a huge difference between the number of positive instances and that of negative instances, the learned model may favor one class as opposed to the other. As a result, the classifier may generate results that look good with respect to classification accuracy, while useless for ranking (e.g. categorizing each

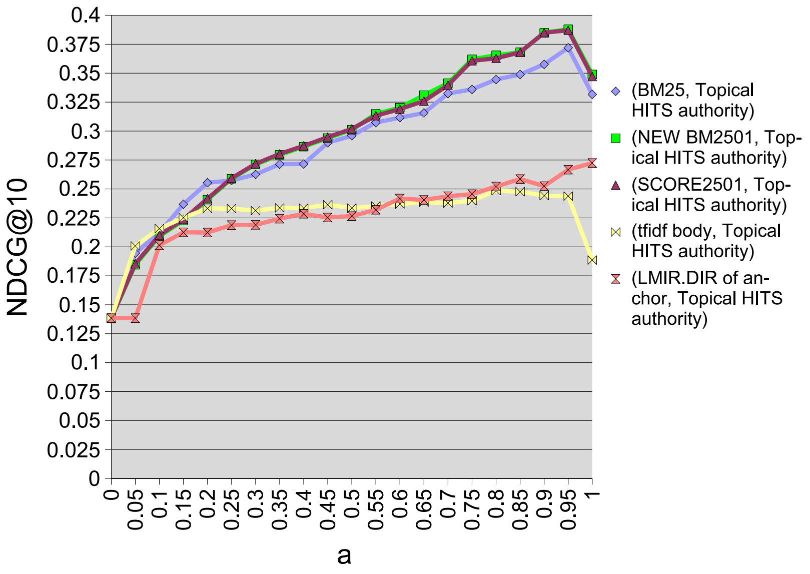

Figure 3: *normRank* of different text-based features combined with Topical HITS authority on TREC2004.

document into the negative class).

- A binary classification model labels each new instance as positive or negative, which is insufficient to generate a ranked list. The question is how to transform a classification result into a ranking. Although this is trivial for algorithms capable of numeric prediction, it is problematic for the other algorithms.
- Even after the second problem is solved so that a score to be used for ranking is generated for each document, it is the ranking that matters in a search engine. A scoring function, optimal for classification accuracy, may not always generate the optimal ranking.

In this study, we focus on the first two problems while leaving the third for future work.

5.1 Unbalanced class distribution

As in previous section, we test our approach on the TREC dataset provided in the LETOR package. As shown in Table 4, there is a dramatic difference between the number of relevant documents and irrelevant documents. Such a difference may pose a danger in the learned model so that when making decisions, the classifier favors the populated class. Preliminary tests confirmed

Dataset	**# relevant documents**	**# irrelevant documents**	**ratio (relevant: irrelevant)**
TREC 2003	516	48,655	1:94
TREC 2004	444	73,726	1:166

Table 4: Number of relevant and irrelevant documents in TREC datasets.

this. A linear kernel SVM classifier trained on the TREC 2003 dataset was tested on the same dataset. Although the training accuracy was high (99%), the recall of positive class (relevant documents) was zero because this classifier categorizes every instance into the negative class.

Since the problem was caused by the extremely unbalanced class distribution, we can alleviate the problem by re-sampling the instances to make the two classes equally represented. One simple solution is to replicate each positive instance n times so that the two classes have equal number of instances. Another solution randomly samples a subset of negative instances while retaining all the positive instances. These simple solutions efficiently alleviate the zero-recall problem. In addition, the first solution, which duplicates positive instances, provides a better precision (3.6% on training set). In the following experiments, the training sets are resampled using the first solution unless stated otherwise.

5.2 Transforming classification results into rankings

The second problem is how to transform the result of binary classification into a ranking. For algorithms that can generate numeric predictions such as support vector machines, this problem can be easily solved by sorting the documents by the predicted values. For other algorithms which are unable to generate such a value, such as decision trees and naive Bayes, we group the documents according to their predicted class, rank the group of predicted relevant documents in front of the group of predicted irrelevant documents, documents in front of all the predicted irrelevant documents, and then use one of the features to decide the ranking inside each group.

We first present in Table 5 the result on TREC of a linear SVM classifier implemented by SVM^{light} [5] using all the 44 features. From the table, we can see that although the resampling decreased classification accuracy, it increased precision of the positive class. Furthermore, it boosted ranking performance significantly. We are also interested in what features are important. In Table 7, we show the 10 most important features according to the model trained on the resampled TREC 2003 data ("+" stands for positive feature, "-" stands for negative feature). We also tested the logistic regression algorithm implemented in Weka [23]. From the result shown in Table 6, there is not a large difference between the resampled test and non-resampled test. A similar experiment using naive Bayes showed that the probabilities generated by naive Bayes classifiers is not a good source for ranking.

Training set	TREC 2003 (not resampled)	TREC 2003 (resampled)	TREC 2004 (resampled)
Test set	TREC 2004	TREC 2004	TREC 2003
Accuracy	99.4%	79.0%	1.1%
Precision	0%	2.4%	1.1%
Recall	0%	86.5%	100%
F-Measure	0	0.047	0.022
MAP	0.092	0.292	0.204
NDCG@10	0.120	0.364	0.274

Table 5: SVM classifiers on TREC.

Training set	TREC 2003 (not resampled)	TREC 2003 (resampled)	TREC 2004 (resampled)
Test set	TREC 2004	TREC 2004	TREC 2003
Accuracy	99.2%	72.5%	85.4%
Precision	16.7%	1.8%	3.5%
Recall	8.3%	84.5%	48.6%
F-Measure	0.111	0.035	0.065
MAP	0.258	0.250	0.120
NDCG@10	0.321	0.306	0.275

Table 6: Logistic regresssion on TREC.

Second, we focus on the classification algorithms that generate nominal (as opposed to numeric) predictions. In this case, based on the classes predicted by the classifier, the documents are grouped into two groups: relevant and irrelevant. A document in the relevant group is always ranked higher than any document in the irrelevant group. The rest of the question is how to decide the ranking inside each group. We propose to use one feature out of the 44 features to decide the intra-group ranking. In our experiment, we se-

Feature Number	**Feature Name**
5 (-)	dl of URL
16 (+)	res-dotGov-anchor.BM25.2003
4 (-)	dl of title
29 (+)	tf of anchor
24 (+)	res-dotGov-ti.MSRAID.BM25.2003
1 (-)	BM25
18 (+)	res-dotGov-anchor.JM.LMIR.2003
42 (+)	UIUCFEAPRO-10-0.9
15 (-)	res-dotGov-anchor.ABS.LMIR.2003

Table 7: Important features of SVM classifiers trained on TREC 2003.

Algorithm	NB	NB	J48	J48	SVM
Training set resampled?	No	Yes	No	Yes	Yes
Accuracy	91%	86%	96%	91%	79%
Precision	5%	3%	2%	1%	2.4%
Recall	75%	85%	11%	16%	86.5%
F-Measure	0.094	0.058	0.034	0.019	0.047
MAP (*CombF42*)	0.294	0.298	0.236	0.160	0.306
NDCG@10 (*CombF42*)	0.358	0.368	0.317	0.178	0.384
MAP (*CombRand*)	0.087	0.072	0.069	0.042	0.051
NDCG@10 (*CombRand*)	0.088	0.065	0.107	0.060	0.044

Table 8: Classifiers without numeric predictions on TREC 2004.

lected the best single feature in terms of NDCG@10, i.e., the 42nd feature, UIUCFEAPRO-10-0.9, with NDCG@10 on TREC'04 using this feature being 0.362, MAP being 0.289. We used naive Bayes (NB) and J48 algorithms implemented by Weka [23], trained on TREC 2003 and tested on TREC 2004. We also compared the above approach, which uses Feature 42 (*CombF42*) to decide intra-group ranking, with the approach which uses a random variable to decide intra-group ranking(*CombRand*). The result of *CombRand* reported in Table 8 is the average of five runs.

From Table 8, we can see that while training set resampling does not have much effect on the naive Bayes classifier, it decreases the performance of J48 with respect to both classification and ranking. In addition, the combination with Feature 42 is significantly better than combining with the random ordering. We also tested the SVM classifier as a non-numeric classifier. The prediction of SVM is nominalized, disregarding the numeric value, and then combined with Feature 42 or a random variable, respectively, as described above. The ranking of *SVM+CombF42* is better than SVM and Feature 42 individually. It also outperforms the combinations of the other classification algorithms with Feature 42.

5.3 Feature selection

In the previous classification experiments, we used all 44 features. In this subsection, we focus on the impact of feature selection on classification and ranking.

One approach to feature selection reduces the dimensionality of the feature space by selecting the most discriminative features, which leads to a reduction in computational complexity of both time and space. Furthermore, a good feature selection algorithm can boost the classification result.

Feature set	M-14	M-text	M-link	A-17
Accuracy	69.0%	0.7%	99.21%	0.7%
Precision	1.6%	0.6%	1.4%	0.6%
Recall	84.5%	100%	0.5%	100%
MAP	0.18	0.05	0.10	0.08
NDCG@10	0.23	0.06	0.13	0.11

Table 9: Experimental results on subsets of features for TREC 2004 queries.

In this study, we performed experiments on the following subsets of features.

- Manual-14 (M-14): the 14 manually selected features used in the arithmetic combinations in the previous section.
- Manual-IR (M-text): the 29 text-based features selected manually.
- Manual-link (M-link): the 13 link-based features selected manually.
- Automatic-17 (A-17): the 17 features automatically selected using *CfsSubsetEval* attribute evaluator and *BestFirst* search method in Weka [23]. "*CfsSubsetEval* assesses the predictive ability of each attribute individually and the degree of redundancy among them, preferring sets of attributes that are highly correlated with the class but have low intercorrelation." [23]

We tested these feature subsets by training SVM classifiers on the TREC 2003 data and testing on TREC 2004 queries. The results are listed in Table 9. Compared with the results based on all 44 features in Table 5, none of these subsets of features improved ranking performance.

6 Ranking by arithmetic combination and classification

In the previous sections, we presented the result of ranking by arithmetic combination and ranking by classification, respectively. In this section, we combine these two approached to find out whether it provides any further improvement.

We take the numeric prediction provided by SVM classifiers as a feature, and combines it with each of the 14 features as listed in Table 1. From the result shown in Table 10, we can see that the combination of arithmetic combination and classification can bring further improvement to ranking. We also tested combining the classifiers listed in Table 8 with each of the 14 features. The performance is significantly worse than the above SVM combination.

7 Discussion and Conclusion

We have compared the TREC topical distillation MAP performance reported by a number of recent publications. In Table 11 we can see that our best performance is 22% better the best of others on TREC2003. Our best performance

Training set	TREC03	TREC03	TREC04	TREC04
Test set	TREC04	TREC04	TREC03	TREC03
Combination Method	NormF1	Norm-Rank	NormF1	Norm-Rank
Best MAP	0.368	0.359	0.236	0.260
Best NDCG@10	0.429	0.430	0.320	0.349
Best when combined with	F_{13}/F_{27}	F_{13}	F_{37}	F_{37}

Table 10: Experiments of ranking by arithmetic combination and classification.

	TREC2003	**TREC2004**
Song et al. [21]	0.212	0.178
Plachouras et al. [14]		0.2785
Qin et al. [15]	0.1719	˜0.16
Xue et al. [24]	0.2022	0.1841
Nie et al. [12]	0.174	
This work	0.260	0.368

Table 11: Comparison of published MAP performance on TREC 2003 and 2004 topic distillation queries.

on TREC2004 is 33.8% better than the best of the others. Even without classification, we have found methods and combinations that can achieve better performance than known previous work.

In summary, we proposed two approaches to combining multiple sources of rankings: arithmetic combination and combination via classification, both of which can individually bring improvement to ranking. In addition, the combination of both approaches is able to improve performance further. In arithmetic combination, we found that variations of the common weighted sum approach were indeed the best performing, and that the combination of text-based and link-based rankers ranked consistently well in both datasets used in our experiment. More generally, simple arithmetic feature combinations were able to achieve significant improvement over any single feature ranker.

While studying combination via classification, we learned that resampling the training set to make both relevant class and irrelevant class equally represented is very useful for SVM classifiers. The effect of such a resampling method on other classifiers is not clear. We also found out that what appears to be a low performing classifier with respect to classification may perform reasonably in a ranking system. Although the classifier did not do well in modeling the boundary of classes, it is still able to learn what patterns make a document more likely to be relevant. As a result, its numeric prediction is enough to generate a reasonable ranking. We also want to point out that the incomplete labeling of the TREC datasets contributes to the difficulty of the learning.

Follow-on work might consider:

- more arithmetic combining functions (such as voting using the top k rather than all);
- using better classification algorithms (such as non-linear kernel SVMs) and meta learners (such as bagging and stacking);
- finding feature selection schemes that are optimized for ranking as opposed to categorization.

Acknowledgments

This material is based upon work supported by the National Science Foundation under grants IIS-0328825 and IIS-0545875.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, New York, May 1999.

[2] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 173–181, 1994.

[3] Google, Inc. Google information for webmasters. Retrieved 9 November 2005 from the Google Website: http://www.google.com/webmasters/4.html, 2005.

[4] S. Hansell. Google keeps tweaking its search engine. *New York Times*, June 2007.

[5] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[6] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–622, 1999.

[7] J. Lafferty and C. Zhai. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of Annual International ACM SIGIR Conference*, pages 334–342, 2001.

[8] G. Lebanon and J. D. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 363–370, San Francisco, CA, USA, 2002. Morgan Kaufmann.

[9] J. H. Lee. Analysis of multiple evidence combination. In *Proceedings of 20th Annual International ACM SIGIR Conference*, pages 267–276, 1997.

[10] D. Lillis, F. Toolan, R. Collier, and J. Dunnion. Probfuse: a probabilistic approach to data fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 139–146, New York, NY, USA, 2006. ACM Press.

[11] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, T. Wang, and H. Li. Letor: Benchmark datasets for learning to rank, 2007.

[12] L. Nie, B. Davison, and X. Qi. Topical link analysis for web search. In *Proceedings of Annual International ACM SIGIR Conference*, 2006.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Unpublished draft, 1998.

[14] V. Plachouras, B. He, and I. Ounis. University of Glasgow at TREC2004: Experiments in web, robust and terabyte tracks with Terrier. In *Proceedings of the 13th Text REtrieval Conference (TREC)*, 2004.

[15] T. Qin, T. Y. Liu, X. D. Zhang, Z. Chen, and W. Y. Ma. A study of relevance propagation for web search. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[16] J. Rapela. Automatically combining ranking heuristics for HTML documents. In *Proceedings of the 3rd International Workshop on Web Information and Data Management (WIDM)*, pages 61–67, New York, 2001. ACM Press.

[17] M. E. Renda and U. Straccia. Web metasearch: Rank vs. score based rank aggregation methods, 2003.

[18] S. E. Robertson. Overview of the okapi projects. *Journal of Documentation*, 53(1):3–7, 1997.

[19] A. Shakery and C. Zhai. Relevance propagation for topic distillation: UIUC TREC 2003 web track experiments. In *Proceedings of TREC*, 2003.

[20] J. A. Shaw and E. A. Fox. Combination of multiple searches. In *Proceedings of the 2nd Text REtrieval Conference (TREC-2)*, pages 243–252, 1994.

[21] R. Song, J.-R. Wen, S. Shi, G. Xin, T.-Y. Liu, T. Qin, X. Zheng, J. Zhang, G. Xue, and W.-Y. Ma. Microsoft research asia at web track and terabyte track of trec 2004. In *Proceedings of the 13th Text REtrieval Conference (TREC2004)*, 2004.

[22] E. Voorhees, N. K. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, 1995.

[23] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, June 2005.

[24] G.-R. Xue, Q. Yang, H.-J. Zeng, Y. Yu, and Z. Chen. Exploiting the hierarchical structure for link analysis. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, New York, 2005. ACM Press.