DYNAMIC DATA FUSION

Ted Diamond
Elizabeth D. Liddy
TextWise LLC
2-212 Center for Science and Technology
Syracuse, NY 13244
liz@textwise.com, ted@textwise.com
Phone: (315) 443-1989

INTRODUCTION

Information retrieval researchers have long appreciated the value of combining, or *fusing*, multiple retrieval systems' relevance scores for a set of documents to improve retrieval performance. However, it is only recently that researchers have begun to consider adjusting the score fusion method to the user's topic and initial results. This study explores the value of fusing multiple retrieval systems' scores in a manner that adjusts to: the semantic and syntactic features of the user's natural language query, the various systems' biases toward long or short documents, and the extent to which the scores produced by the multiple systems are statistically independent.

PREVIOUS WORK

The ability to improve retrieval performance by using multiple retrieval systems has been documented extensively (e.g., [1], [3], [4]). It is only recently, however, that researchers have turned their attention to the possibility of adjusting the manner in which results are combined to the specific query at hand. Researchers have reported success in using initial relevance judgments to adjust the *way* in which results are combined [3], and have also reported success in using the joint distribution of relevance scores from multiple matchers (among other things) to predict *when to combine* the results of multiple systems [6]. The purpose of the current research is to explore the use of the joint distribution of relevance scores, semantic and syntactic features of queries, and the length of retrieved documents to predict *how to combine* the results of several retrieval systems.

DEFINITIONS AND RESEARCH QUESTIONS

We define a *query* as a natural language expression of a user's need. For some query and some collection of documents, it is possible for a human to attribute the *relevance* of the document to the query. A *retrieval system* is a machine that accepts a query and full texts of documents, and produces, for each document, a *relevance score* for the query-document pair. A measure of the effectiveness of a retrieval system for a query and a collection is *precision*, the proportion of the N documents with the highest relevance scores that are relevant (in our study, N is 5, 10, or 30).

Using multiple retrieval systems produces multiple retrieval scores for a query-document pair. A *fusion function* accepts these scores as its inputs, and produces a single relevance score as its output for the query-document pair. A *static fusion function* has *only* the relevance scores for a single query-document pair as its inputs. A *dynamic fusion function* can have more inputs.

We are concerned with the following two questions:

1. If we allow each query its own static fusion function, can we achieve higher precision than if we force all queries to have the same static fusion function?

2. If we can achieve higher precision by allowing each query its own *static* fusion function, then what inputs or features would enable us to construct a *dynamic* fusion function that adjusts to the query, the documents retrieved by the retrieval systems, and the distribution of scores produced by the retrieval systems?

THE DATA

Queries, Documents, and Relevance Judgments

We used 247 queries, including TREC 1-6 training queries, and queries developed by business analysts for TextWise's internal use. We applied these queries to the TREC Wall Street Journal collection from (1986-1992). For the TREC queries, we used only TREC relevance judgments. Relevance judgments for the TextWise queries were initially made on a 5-point scale, which we mapped to the binary judgments used by TREC.

Several of the retrieval systems described below used a document segmentation scheme to split compound documents into their components, resulting in a collection size of 222,525. For these systems, retrieval scores were calculated separately for the components of compound documents, and then merged by taking the maximum component score, thus mapping back to the original document space of 173,252.

Retrieval Systems

We used five retrieval systems to generate relevance scores for query-document pairs:

Fuzzy Boolean (FB). This system translates a query into a Boolean expression in which the terminals are single terms, compound nominals, and proper nouns; instantiates the terminals in the expression with the document's *tf.idf* weights; and applies fuzzy Boolean semantics to resolve the instantiated expression into a scalar relevance score.

Probabilistic (PRB). This system applies a match formula that sums term frequencies of query terms in the document, weighted by terms' inverse document frequencies, and adjusts for document length. We applied this formula to a vocabulary of single terms.

Subject Field Code (SFC). This system applies a vector similarity metric to query and document representations in TextWise's Subject Field Code space to obtain relevance scores.

N-gram (NG3). This system applies a vector similarity metric to query and document representations obtained by counting the occurrences of 3-letter sequences (after squeezing out blanks, newlines, and other non-alphabetic characters).

Latent Semantic Indexing (LSI). This system obtains query and document representations by applying a translation matrix to single terms (excluding compound nominals and proper nouns). We obtained the translation matrix by singular value decomposition of a matrix of *tf.idf* weights for single terms from a 1/3 sample of the Wall Street Journal. We used a vector similarity metric to obtain relevance scores.

Query and Document Representations

We used the following procedures to process the queries and documents into forms that enabled application of matching formulae to produce relevance scores:

Document Segmentation. We used either the original document segmentation from the TREC data or a more aggressive segmentation that split compound documents into their components.

Stop Word Removal. For all but one retrieval system, we removed stopwords.

Stemming. For the various retrieval systems, we used the Xerox stemmer, the Stone stemmer, or we obtained word roots as a byproduct of constructing trigrams.

Phrase Recognition. For some retrieval systems, we used a set of part-of-speech-based rules to detect and aggregate sequences of tokens into compound nominal phrases.

Proper Nouns. For some retrieval systems, we detected proper nouns, and normalized multiple expressions of the same proper noun entity to a canonical form.

Term Weighting. In documents, weights represented the frequency of terms in the document, conditioned by the number of documents in which the terms

Table 1
Features of Retrieval Systems

	Retrieval Systems				
FEATURE	FB	PRB	SFC	NG3	LSI
Doc. Segmentation	Aggressive	Aggressive	Aggressive	Standard	Aggressive
Stop Word Removal	Yes	Yes	Yes	No	Yes
Stemming	Xerox	Xerox	Stone	Trigram	Xerox
Phrase Recognition	Yes	No	No	No	No
Proper Nouns	Yes	Yes	No	No	No
Term Weighting	*tf.idf*	*tf.idf*	*tf*	*tf.idf*	*tf.idf*
Dimension Reduction	None	None	SFC	None	LSI
Match Semantics	Fuzzy Boolean	Probabilistic	Vector	Vector	Vector

appeared (*tf.idf*).

Dimension Reduction. We used single words to translate into weightings in a 900-dimensional feature space using TextWise's Subject Field Coder (SFC), or into a 167-dimensional feature space using Latent Semantic Indexing (LSI).

Table 1 summarizes the query representations, document representations, and matching semantics used by the five matchers.

Dynamic Fusion Function Input Features

In addition to the five relevance score inputs to the dynamic fusion function, we used the following inputs:

Query Features

Several items of information might be available about the query independently of any particular retrieval approach or its representation of the query, the documents, or their similarity:

Query Length (QLEN). The number of tokens in the natural language query.

Query Terms' Specificity (QTSP). The average inverse document frequency (IDF) of the quartile of the query's terms with the highest IDF's.

Number of Proper Nouns (QNPN).

Number of Compound Nominals (QNCN).

Query Terms' Synonymy (QTSY). Over all terms in the query, the average of the number of words in the *synset* for the correct sense of the query term in WordNet. WordNet is a semantic knowledge base that distinguishes words by their senses, and groups *word:senses* that are synonymous to each other into synsets.

Query Terms' Polysemy (QTPL). Over all terms in query, the average number of senses for the query term in WordNet.

Document Features

There is currently one document feature, instantiated separately for each query, for each retrieval system S:

Length of Top-Ranked Documents Retrieved by System (DLEN[S]). This is the average of the number of tokens in the top 5 documents scored by system S.

Score Distributions

The following features are instantiated once for each retrieval system S, for each query:

Maximum Score Assigned by Approach (SMAX[S]).

Variance of Scores Assigned by Approach (SVAR[S]).

The following input to the dynamic fusion function is instantiated once for each pair of retrieval systems S_1 and S_2:

Correlation of Ranks Assigned to Documents by Two Approaches (SCOR[S_1, S_2]). For documents ranked in the top 1,000 by *any* of the retrieval systems for the query, the correlation of the documents' ranks in systems S_1 and S_2.

RESEARCH QUESTION 1: OPPORTUNITY FOR IMPROVING RETRIEVAL

For a sample of 50 queries from our 297, we found, separately for each query, an optimal static fusion function. We then found the single optimal static fusion function that gave the best precision over all 50 queries. Table 2 shows the precision for the 50 queries using the 5 retrieval systems

Table 2.
Precision of Five Systems, Overall Static Fusion Functions, and Query-Specific Fusion Functions, When Training and Testing on Same Data for Each Query

	Single Retrieval Systems					Static Fusion Functions	
Prec. at	FB	SFC	PROB	NG3	LSI	Single Overall (vs. FB)	Query-Specific (vs. overall)
5	.3360	.0080	.2080	.1760	.1640	.3840 (+14%)	.5960 (+55%)
10	.2680	.0060	.1800	.1560	.1440	.3280 (+22%)	.5040 (+54%)
30	.2240	.0127	.1193	.1414	.1273	.2547 (+14%)	.3427 (+35%)

separately, using a single overall static fusion function, and using 50 (possibly) different query-specific static functions.

At first glance, our results suggest that allowing query-specific fusion functions substantially improves retrieval. For instance, by using query-specific static fusion functions, we achieved precision at 5 of .5960, compared to .3840 when applying the same static fusion function to all queries. However, this comparison is overly optimistic, since it allows query-specific fusion functions to be trained and evaluated on exactly the same data, while forcing the overall fusion function to be trained on a large set of data, but then evaluated on a small subset of that data. To provide a more pessimistic comparison, we partitioned the data for our 50 queries into equally-sized training and test sets. We trained each query-specific fusion function on the query's training data, and evaluated it on the test data. (Although our goal is to improve retrospective retrieval, this arrangement resembles the TREC routing scenario.) Table 3 shows a considerably weaker, but still appreciable improvement due to using query-specific fusion functions. For instance, we achieved precision at 5 of .4160 when allowing each query its own static fusion function, compared to .3400 when forcing all queries to use the same function.

dimensions are the relevance scores from the set of matchers. We constructed a fused score for a test document by summing the relevance judgments for the test document's K nearest training documents (where K was 5, 10, 15 or 20). We tried weighting the sums by an inverse function of the distance between the test document and the training document. We also tried scaling the dimensions' contribution to the distance metric with a weight reflecting the corresponding matcher's precision.

To our surprise, none of these experiments produced K-NN-based fusion functions that performed consistently better than a linear fusion function. On closer inspection, it appears that at least part of the poor performance of K-NN as a fusion function can be attributed to instances in which the probability distribution of relevance for the training documents for the query did not resemble the probability distribution of relevance for all the documents in the query. In this sort of situation, the linear model appears to be more robust than K-NN. It may be that a more careful selection of the training set would result in more reasonable performance from K-NN-based fusion functions.

For the linear fusion function, we found the optimal vector of coefficients by selecting the coefficients that produce the greatest precision at 5

Table 3.
Precision of Five Systems, Overall Static Fusion Functions, and Query-Specific Fusion Functions, When Training and Testing on Different Data for Each Query

	Single Retrieval Systems					Static Fusion Functions	
Prec. at	FB	SFC	PROB	NG3	LSI	Single Overall (vs. FB)	Query-Specific (vs. overall)
5	.3360	.0040	.1960	.1920	.1600	.3400 (+01%)	.4160 (+22%)
10	.2680	.0100	.1680	.1600	.1340	.3120 (+16%)	.3620 (+16%)
30	.1967	.0140	.1187	.1313	.1253	.2230 (+13%)	.2533 (+34%)

We constrained our fusion functions to be weighted *linear* combinations of the five retrieval scores for a query-document pair. We considered the possibility of more complex non-linear fusion models through exploration of K-Nearest Neighbor (K-NN) classifiers. (The use of K-NN for selecting a single retrieval system has been documented in [5]. By contrast, we sought to use K-NN to fuse relevance scores.) In this approach, training documents and their relevance judgments populated a space whose

(the proportion of the five top-ranked documents that are relevant). To date, we have found the optimal vector using an exhaustive search over the set of vectors whose elements are non-negative, evenly divisible by 0.1, and whose elements sum to 1.0. (We had tried using logistic regression to find the coefficients, but the coefficients we found in this manner yielded considerably lower precision than those we found using the exhaustive search method.)

In sum, it appears that for our selection of retrieval systems, there is a *potential* for improving retrieval through query-specific fusion.

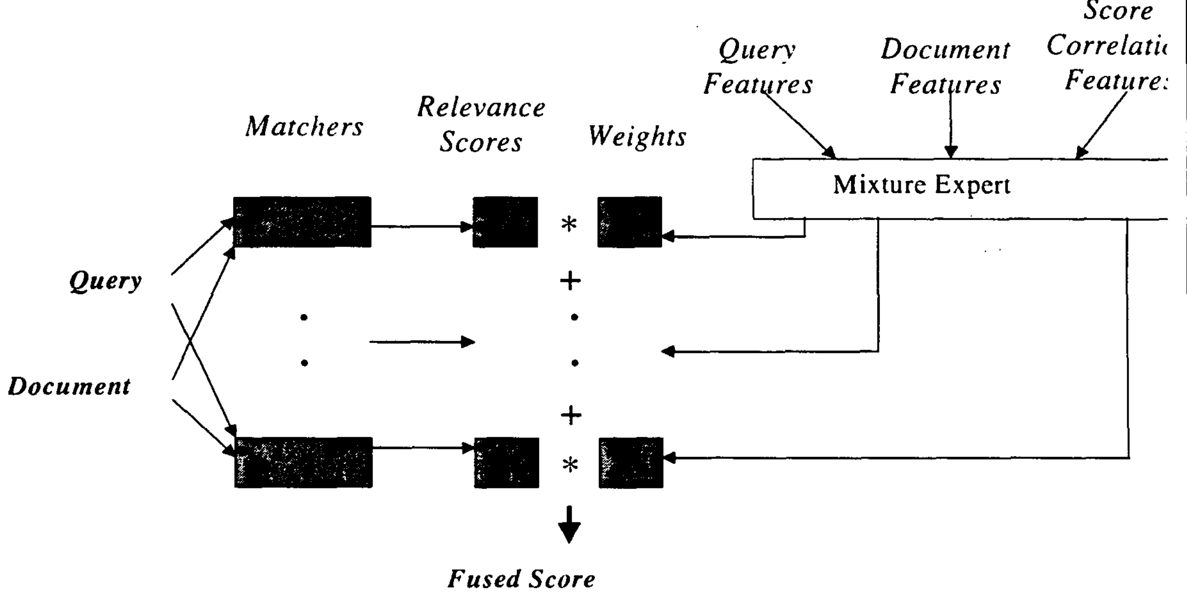
One way to exploit this opportunity is to use initially-retrieved documents to adjust the weights of the single overall static fusion function, as in [3]. Although we tried several ways of updating fusion function coefficients with relevance feedback, we were unable to exploit any of the apparent potential to improve retrieval performance in this way.

distribution of the retrieval systems' retrieval scores for the query enumerated above. We are currently working on building such a dynamic fusion function.

Dynamic Fusion Function Architecture

We chose to implement the dynamic fusion function as a hybrid of a "mixture expert" and the static linear fusion models used in Research Question 1. The mixture expert attempts to predict the best coefficients to use for the linear fusion function. Figure 1 shows the relationship of the mixture expert

Figure 1
Dynamic Fusion Functions



to the linear fusion model and the individual retrieval systems.

RESEARCH QUESTION 2: THE DYNAMIC FUSION FUNCTION

So far, optimal fusion coefficients for a query have been determined using full knowledge of the relevance of the documents for the query. In the retrospective retrieval setting, these relevance judgments will not be available beforehand, and thus cannot be used to adjust the fusion model to the query. For the retrospective setting, we seek to construct a dynamic fusion function that can adjust the way it fuses the five systems' relevance scores for a query-document pair using additional inputs. These inputs include the features of the query, features of the retrieved documents, and features of the joint

Training and Evaluation

We use the remaining 197 queries for training. For these queries, we have used all the documents to find coefficient vectors for optimal linear static fusion models. These coefficient vectors constitute the "target" outputs the mixture expert will be trained to reproduce.

We also fit a single linear static fusion function to the 197 training queries, again using all the data from those queries. The performance of this static fusion function on all of the documents for the 50 test queries constitutes the baseline for the second research question. To answer this research question, we will compare the performance of the dynamic fusion function for the 50 test queries to this baseline.

Discussion

So far, our results suggest that, for our choice of retrieval systems, there *is* an opportunity to improve retrieval performance by using dynamic fusion functions instead of using a single static fusion function for all queries. One possible qualification to these results is that limiting ourselves to a linear form for the static fusion models may result in artificially low baseline retrieval for the single overall static function. The volatility of the *K*-NN technique in the context of our data made it difficult to say whether or not a non-linear form for the fusion model is necessary.

Our preferred implementation of the mixture expert in the dynamic fusion function is a multilayer feedforward neural network, with output nodes corresponding to the linear weights of the linear fusion function. However, given that our real goal is to maximize precision, rather than to replicate the weights exactly, a straightforward application of backpropogation to train such a network to replicate the target weights is inappropriate. The optimal linear weights are likely to be on "plateaus" with respect to precision, with little change in precision in response to large changes in linear weights. We are currently investigating alternative ways of training the mixture expert in the dynamic fusion model.

References

[1] Bartell, B., Cottrell, G.W., Belew, R.K. Automatic combination of multiple ranked systems. Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, 173-181, 1994.

[2] Belkin, N., Kantor, P., Fox, E., Shaw, J.. Combining the evidence of multiple query representations for information retrieval. Information Processing and Management 31(3), 431-448, 1995.

[3] Fox, E., Shaw, J. Combination of multiple searches. In The Second Text Retrieval Conference (TREC-2), D. Harman (ed), NIST Special Publications 500-215, Gaithersburg, MD, 242-252, 1994.

[4] Hull, D., Pedersen, J., Schuetze, H. Method combination for document filtering. Proceedings of the 19th Annual Internation ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, 279-288, 1996.

[5] Savoy, J., Ndarugendawmo, M., Vrajitoru, D. Report on the TREC-4 experiment: combining probabilistic and vector-space schemes. [TREC-4 WWW site], 1996.

[6] Vogt., C., Cottrell, G.W. Predicting the Performance of Linearly Combined IR Systems. Proceedings of the Twenty First Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, 1998.