Combining Multiple Evidence from Different Properties of Weighting Schemes

Joon Ho Lee*
Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

Abstract

It has been known that using different representations of either queries or documents, or different retrieval techniques retrieves different sets of documents. Recent work suggests that significant improvements in retrieval performance can be achieved by combining multiple representations or multiple retrieval techniques. In this paper we propose a simple method for retrieving different documents within a single query representation, a single document representation and a single retrieval technique. We classify the types of documents, and describe the properties of weighting schemes. Then, we explain that different properties of weighting schemes may retrieve different types of documents. Experimental results show that significant improvements can be obtained by combining the retrieval results from different properties of weighting schemes.

1 Introduction

A variety of representation techniques for queries and documents have been proposed in the information retrieval literature, and many corresponding retrieval techniques have also been developed to get the higher effectiveness of information retrieval. Recent research shows that retrieval effectiveness can be improved by using multiple query or document representations, or multiple retrieval techniques, and combining the retrieval results, in contrast to using just a single representation or a single retrieval technique. This general area has been discussed in the literature under the name of "data fusion".

McGill, Koll & Norreault [1] found that there was surprisingly little overlap between document sets for the same information need, when documents were retrieved by different users or by the same user using controlled versus free-text vocabularies. Katzer, et al. [2] considered the effect of different document representations, e.g. title, abstract on retrieval effectiveness rather than different query representations. They discovered the same phenomenon that the various document representations gave similar retrieval effectiveness, but retrieved quite different sets of documents. These results

*On leave from Korea Research and Development Information Center, Korea Institute of Science and Technology, P.O. Box 1, Yusung, Taejon 305-600, Korea

suggest that the combined run may retrieve more relevant documents than any individual run, therefore providing higher recall.

Saracevic and Kantor [3] asked different experts to construct Boolean queries based on the same description of information problem in operational online information retrieval systems. Once again, they found that different query formulations generated different documents. They, however, noticed that the odds of a document being judged relevant increase monotonically with the number of retrieved sets in which the document appears. If the combining method is designed to favor the documents retrieved by more retrieval runs, the combined run can result in more accurate similarity values between queries and documents, and therefore give higher precision.

Turtle and Croft [4] developed an inference network-based retrieval model to combine different document representations and different versions of a query in a consistent probabilistic framework. The network model treats different representations as evidence that is combined to estimate the probability of a document satisfying a user's information need. Turtle and Croft implemented the INQUERY retrieval system based on the model, and demonstrated that multiple evidence increases retrieval effectiveness in some circumstances. Fox and Shaw [5] have worked on various methods for combining multiple retrieval runs, and have obtained improvements over any single retrieval run. Belkin, et al. [6] showed that progressive combination of different Boolean query formulations could lead to progressive improvements of retrieval effectiveness.

The research results described above show that combining multiple retrieval runs can improve the effectiveness of information retrieval. However, only multiple query or document representations, or multiple retrieval techniques have been taken into consideration to generate multiple retrieval runs. Harman [7] observed that the systems of the first Text REtrieval Conference (TREC-1) retrieved substantially different sets of documents, even though many systems performed at the same approximate level. This observation suggests that there may be other ways of retrieving different documents and improving retrieval effectiveness.

In this paper we propose a method for retrieving different sets of documents within just a single query representation, a single document representation and a single retrieval technique. We classify document types depending on the number of topics described by documents and the number of indexable words in documents. We describe the properties of document weighting schemes, i.e. cosine normalization and maximum normalization, and explain how they affect the document types retrieved. We also explain that different documents may be retrieved by the weighting schemes having different properties. Experimental results show that whether cosine normalization is used or not plays an important role of retrieving different sets of documents. We also show through experiments that significant improvements can be obtained by combining the two retrieval runs in which one performs cosine normalization and the other does not.

The remainder of this paper is organized as follows. Section 2 gives the description of the SMART system that is used to evaluate our runs. In section 3 we classify document types and analyze the properties of weighting schemes, and we show that different documents can be retrieved by different properties of weighting schemes. Section 4 shows through experimental results that significant improvements can be achieved by the combination of the two runs using different properties of weighting schemes. Finally, the summary and future works are given in section 5.

2 The SMART System

The SMART system [8] has been developed at Harvard and Cornell Universities for over 30 years. The indexing of both queries and documents is completely automatic, and therefore human experts are not required for either the initial collection creation or the actual query formulation. This means that retrieval results are reasonably collection independent and should be valid across a wide range of collections.

2.1 Similarity computation

SMART is based on the vector space model [9], and transforms the description of information problems as well as the stored documents into vectors of the form:

$$d_i = (w_{i1}, w_{i2}, \ldots, w_{in})$$

where d_i represents a document (or query) text and w_{ik} is the weight of term t_k in document d_i. The assumption is that n terms in all are available for the representation of queries and documents. A weight of zero is used for terms that are absent from a particular document, and positive weights characterize terms actually assigned for content identification. In the SMART context, such vectors are formed by a text transformation as follows:

1. recognize individual text words
2. eliminate function words with a stop list
3. generate word stems by removing suffixes
4. assign term weights to all remaining word stems to form the term vector

Even though SMART can handle term phrases formed with statistical word co-occurrence or syntactic analysis, we will be concerned only with word stems in generating query and document vectors in this paper.

Once term vectors are available for all information items, all subsequent processing is based on term vector manipulations. When document d is represented by a vector of the form (w_{d1}, w_{d2}, ..., w_{dn}) and query q by the vector (w_{q1}, w_{q2}, ..., w_{qn}), the similarity between document d and query q is calculated as the inner product between corresponding weighted term vectors as follows:

$$Sim(d, q) = \sum_{i=1}^{n} (w_{di} \times w_{qi})$$

The query-document similarity depends on the weights of coinciding terms in the two vectors, and therefore the term weighing scheme is an important factor affecting the effectiveness of SMART.

2.2 Term weighting schemes

In constructing a term weighting scheme, three main components such as term frequency, collection frequency and normalization have been considered in the information retrieval literature [10]. First,

Table 1: Components of term weighting schemes

Term Frequency Component		
b	1.0	binary weight equal to 1 for terms present in a vector (term frequency is ignored)
n	tf	raw term frequency (number of times a term occurs in a document or query text)
a	$0.5 + 0.5\frac{tf}{\max tf}$	augmented normalized term frequency (use maximum normalization where each tf is divided by maximum tf, and further normalize the resulting value to lie between 0.5 and 1.0)
l	$\ln tf + 1.0$	logarithmic term frequency which reduces the importance of raw term frequency in those collections with widely varying document length
Collection Frequency Component		
n	1.0	no change in weight; use original term frequency component (b, n, a, or l)
t	$\ln \frac{N}{n}$	multiply original term frequency component by an inverse document frequency factor (N is the total number of documents in the collection, and n is the number of documents to which a term is assigned)
Normalization Component		
n	1.0	no change; use factors derived from term frequency and collection frequency only (no normalization)
c	$\frac{1}{\sqrt{\sum_{vector} w_i^2}}$	use cosine normalization where each term weight w_i is divided by a factor representing Euclidian vector length

the term frequency component assigns higher weights to the terms that occur more frequently in the text. Second, the collection frequency component assigns higher weights to the terms that occur in fewer documents of the collection. The normalization component equalizes the length of document vectors in the collections with varying document vector length.

Table 1 shows actual formulas for some well-known term weighting schemes. A term weighting scheme is described by using two triples representing the term frequency, collection frequency and normalization. The first and second triples are for document terms and query terms, respectively. For instance, the $lnc \cdot ltc$ system, which gives high retrieval effectiveness for the TREC data collections, uses cosine normalization of logarithmic term frequency for document term weights, and cosine normalization of logarithmic term frequency $\times$ inverse document frequency for query term weights.

The effectiveness of a retrieval system is much dependent on the weighting scheme used in the system. We classify weighting schemes into two groups depending on the normalization component, and evaluate their effectiveness with one of the TREC subcollections, namely the Wall Street Journal Disk 2 (WSJ.D2) [11]. We retrieve the top-ranked 200 documents for 100 queries, and eval-

Table 2: 11-point average precision for the weighting schemes with cosine normalization (WSJ.D2 collection: averages over 100 queries)

	document weighting	
	lnc	*anc*
query weighting		
btc	0.2739	0.2457
ntc	0.3103(2)	0.2836 (6)
atc	0.3101(3)	0.2862 (5)
ltc	0.3284(1)	0.3034 (4)

Table 3: 11-point average precision for the weighting schemes without cosine normalization (WSJ.D2 collection: averages over 100 queries)

	document weighting			
	lnn	*ann*	*ltn*	*atn*
query weighting				
btc	0.1567	0.1846	0.2148	0.2265
atc	0.1885	0.2167	0.2493	0.2608
ltc	0.2170	0.2518	0.2749(5)	0.2877(4)
ntc	0.2661(6)	0.3002(3)	0.3028(2)	0.3198(1)

uate the performance using the 11-point average precision. The results presented in Table 2 and 3 show that different weighting schemes provide quite different retrieval effectiveness. The number in parentheses is the rank of the weighting scheme in each group of weighting schemes. $lnc \cdot ltc$ provides the best retrieval effectiveness in the weighting schemes with cosine normalization, and $atn \cdot ntc$ in those without cosine normalization.

3 Analyzing Weighting Schemes

Turtle and Croft [4] evaluated both probabilistic and Boolean versions of the query, and combined the results. Combining queries resulted in significant performance improvements for the CACM and CISI collections. They also gave an interesting analysis, which is quoted in the following:

> We originally thought that at least part of the performance improvements arose because the two query types were retrieving different relevant documents, so that the combined set contained more relevant documents than retrieved by the separate queries. This is not, however, the case. The documents retrieved by the Boolean queries are a subset of those retrieved by the corresponding probabilistic query.
>
> The Boolean queries in the test set do not add new terms or domain knowledge. Performance improvements appear to arise because the Boolean queries capture structural information in the queries (phrase structure, compound nominals, and negation) that is not exploited in probabilistic queries.

Table 4: Statistics of TREC document collections

Subset of Collection	WSJ(disks 1 and 2) SJMN (disk 3)	AP	ZIFF	FR(disks 1 and 2) FAT(disk 3)	DOE
median number of words per record					
(disk1)	182	353	181	313	82
(disk2)	218	346	167	315	
(disk3)	279	358	119	2896	
average number of words per record					
(disk1)	329	375	412	1017	89
(disk2)	377	370	394	1073	
(disk3)	337	379	263	3543	

The above analysis suggests that Turtle and Croft could not get different documents even though they used different query formulations such as Boolean and probabilistic queries. Furthermore, they could not retrieve different documents with the different runs using the same document and query terms. However, there are a variety of formulas, e.g. $lnc \cdot ltc$, $atn \cdot ntc$, et al. to weight document and query terms. In the following sections, we explain that different documents can be retrieved by applying different properties of weighting schemes to a single document and query formulation.

3.1 Classification of document types

In what follows, we define *tf-vector length* as the summation of term frequencies, i.e. $\sum_{i=1}^{n} tf_i$ where tf_i is the frequency of term t_i and n is the number of terms representing document vectors. For example, when document d_1 is represented by pairs of a term and its frequency as follows:

$$d_1 = \{(t_1, 1), (t_2, 2), (t_3, 3), (t_4, 4), (t_5, 5)\}$$

the tf-vector length of document d_1 is equal to $15 (= 1 + 2 + 3 + 4 + 5)$. The indexable words of a document, which are used to form the vector for the document, are the remaining words after eliminating stop words. Since query-document similarities depend on indexable words rather than all of words in original document texts, we focus on the number of indexable words, or, equivalently, tf-vector length rather than the number of words including stop words that is often called *document length*.

Many realistic data collections such as the TREC collections have widely varying document length, which can result in widely varying numbers of indexable words, that is, widely varying tf-vector length. We classify documents into the three types such as the documents of *short*, *median* and *long* tf-vector length. It should be noted that we use median tf-vector length rather than average tf-vector length. We can get median tf-vector length as follows: First, sort records in decreasing order of the number of indexable words in the records. Then, median tf-vector length is the number of indexable words included in the middle record. This means that most documents have median

tf-vector length rather than average tf-vector length, and therefore a weighting scheme should be designed to retrieve median tf-vector length documents of being relevant better than short, long and even average tf-vector length documents.

We also classify documents into the two types consisting of *single* and *multiple* topic documents. Table 4 shows the statistics of the TREC document collections [11]. In many subcollections the average number of words is much greater than the median, which means there are too many long documents. Note that the greater the average is than the median, the more long documents there are. It is known that long documents and even short documents summarizing many subjects deal with multiple topics rather than a single topic in many cases. Then, considerations have encouraged research interests in retrieving parts of documents rather than whole documents [12], and using passage-level evidence to calculate query-document similarities [13].

3.2 Properties of weighting schemes

If we use weighting schemes without tf-vector length normalization, the documents of long tf-vector length have a better chance of being retrieved than those of short tf-vector length. For example, suppose that two documents d_2 and d_3 are represented by pairs of a term and its frequency as follows:

$$d_2 = \{(t_1, 1), (t_2, 1), \ldots, (t_n, 1)\}$$

$$d_3 = \{(t_1, 2), (t_2, 2), \ldots, (t_n, 2)\}$$

If we apply the *lnn* weighting formula, i.e. $\ln tf + 1.0$, which does not have tf-vector length normalization factors, the term weights of $d_{3.lnn}$ are greater than those of $d_{2.lnn}$ as

$$d_{2.lnn} = \{(t_1, 1), (t_2, 1), \ldots, (t_n, 1)\}$$

$$d_{3.lnn} = \{(t_1, 1.69), (t_2, 1.69), \ldots, (t_n, 1.69)\}$$

For query $q_1 = \{(t_1, w_1), (t_2, w_2), \ldots, (t_n, w_n)\}$, the similarities of $d_{2.lnn}$ and $d_{3.lnn}$ are calculated by the inner product as follows:

$$Sim(d_{2.lnn}, q_1) = w_1 + w_2 + \ldots + w_n = \sum_{i=1}^{n} w_i$$

$$Sim(d_{3.lnn}, q_1) = 1.69 \cdot w_1 + 1.69 \cdot w_2 + \ldots + 1.69 \cdot w_n = 1.69 \cdot \sum_{i=1}^{n} w_i$$

Therefore, the *lnn* weighting formula makes the long tf-vector document d_3 be given higher rank than the short tf-vector document d_2, which may not agree with most people's decision. Note that the documents d_2 and d_3 can be considered as almost the same documents if we consider tf-vector length.

As explained above, the absence of tf-vector length normalization factor makes the system favor the documents of longer tf-vector length. All documents should be treated as equally important for retrieval purposes. This suggests that *cosine normalization* be incorporated into the term weighting system of SMART to equalize the length of document vectors. It has been known that weighting

schemes with cosine normalization perform better than those without cosine normalization in many data collections.

Even though cosine normalization has a desirable property of normalizing tf-vector length, it may make it difficult to retrieve relevant documents dealing with multiple topics. This is because the weights of relevant terms are decreased by nonrelevant terms where relevant terms are the terms related with user's information need, or, possibly, those specified in the query. For example, suppose that document d_4 describes a single topic, and document d_5 describes more than two topics including the topic of document d_4. The documents d_4 and d_5 can be represented by pairs of a term and its frequency as follows:

$$d_4 = \{(t_1, 1), \ldots, (t_m, 1), (t_{m+1}, 0), \ldots, (t_n, 0)\}$$

$$d_5 = \{(t_1, 1), \ldots, (t_m, 1), (t_{m+1}, 1), \ldots, (t_n, 1)\}$$

The *lnc* weighting formula converts documents d_4 and d_5 as

$$d_{4.lnc} = \{(t_1, \frac{1}{\sqrt{m}}), \ldots, (t_m, \frac{1}{\sqrt{m}}), (t_{m+1}, 0), \ldots, (t_n, 0)\}$$

$$d_{5.lnc} = \{(t_1, \frac{1}{\sqrt{n}}), \ldots, (t_m, \frac{1}{\sqrt{n}}), (t_{m+1}, \frac{1}{\sqrt{n}}), \ldots, (t_n, \frac{1}{\sqrt{n}})\}$$

For query $q_2 = \{(t_1, w_1), \ldots, (t_m, w_m), (t_{m+1}, 0), \ldots, (t_n, 0)\}$, the similarities of $d_{4.lnc}$ and $d_{5.lnc}$ are calculated by the inner product as follows:

$$Sim(d_{4.lnc}, q_2) = \frac{1}{\sqrt{m}} \cdot w_1 + \frac{1}{\sqrt{m}} \cdot w_2 + \ldots + \frac{1}{\sqrt{m}} \cdot w_m = \frac{1}{\sqrt{m}} \cdot \sum_{i=1}^{m} w_i$$

$$Sim(d_{5.lnc}, q_2) = \frac{1}{\sqrt{n}} \cdot w_1 + \frac{1}{\sqrt{n}} \cdot w_2 + \ldots + \frac{1}{\sqrt{n}} \cdot w_m = \frac{1}{\sqrt{n}} \cdot \sum_{i=1}^{m} w_i$$

Since n is always greater than m, the single topic document d_4 is given higher rank than the multiple topic document d_5 even though documents d_4 and d_5 have the same amount information for query q_2. This undesirable result is due to the fact that the weights of relevant terms t_1 through t_m in document d_5 are decreased by nonrelevant terms t_{m+1} through t_n.

The augmented normalized term frequency denoted as 'a' normalizes the frequency of a term (tf) by the maximum frequency of any term in the document ($\max tf$). This *maximum normalization* can normalize tf-vector length in certain cases. For example, suppose that two documents d_6 and d_7 are represented by pairs of a term and its frequency as follows:

$$d_6 = \{(t_1, 1), (t_2, 1), \ldots, (t_n, 1)\}$$

$$d_7 = \{(t_1, 2), (t_2, 2), \ldots, (t_n, 2)\}$$

If we apply the *ann* weighting formula, i.e. $0.5 + 0.5\frac{tf}{\max tf}$, documents d_6 and d_7 have the same vector representation as

$$d_{6.ann} = \{(t_1, 1), (t_2, 1), \ldots, (t_n, 1)\}$$

$$d_{7.ann} = \{(t_1, 1), (t_2, 1), \ldots, (t_n, 1)\}$$

Maximum normalization, however, cannot normalize tf-vector length in many cases that cosine normalization can. For example, suppose that two documents d_8 and d_9 are represented by pairs of a term and its frequency as follows:

$$d_8 = \{(t_1, 1), (t_2, 1), \ldots, (t_{100}, 1)\}$$

$$d_9 = \{(t_1, 2), (t_2, 1), \ldots, (t_{100}, 1)\}$$

If we apply the *ann* weighting formula, the term weights of $d_{8.ann}$ are greater than those of $d_{9.ann}$ as

$$d_{8.ann} = \{(t_1, 1), (t_2, 1), \ldots, (t_{100}, 1)\}$$

$$d_{9.ann} = \{(t_1, 1), (t_2, 0.75), \ldots, (t_{100}, 0.75)\}$$

Therefore, d_8 has a much better chance of being top-ranked than d_9 even though d_8 and d_9 can be considered as almost the same documents. Note that only the frequency of term t_1 is different between documents d_8 and d_9. On the contrary, if we apply the *lnc* weighting formula, documents d_8 and d_9 are converted as

$$d_{8.lnc} = \{(t_1, 0.1), (t_2, 0.1), \ldots, (t_{100}, 0.1)\}$$

$$d_{9.lnc} = \{(t_1, 0.167), (t_2, 0.099), \ldots, (t_{100}, 0.099)\}$$

Since term weights of $d_{8.lnc}$ are almost equal to those of $d_{9.lnc}$ except the weight of term t_1, documents d_8 and d_9 have a similar level of similarity values with respect to a given query.

While maximum normalization cannot normalize tf-vector length in many cases, it may alleviate the problem of cosine normalization that may make it difficult to retrieve relevant documents dealing with multiple topics. For example, suppose that document d_{10} describes a single topic, and document d_{11} describes more than two topics including the topic of document d_{10}. The documents d_{10} and d_{11} can be represented by pairs of a term and its frequency as follows:

$$d_{10} = \{(t_1, 1), \ldots, (t_m, 1), (t_{m+1}, 0), \ldots, (t_n, 0)\}$$

$$d_{11} = \{(t_1, 1), \ldots, (t_m, 1), (t_{m+1}, 1), \ldots, (t_n, 1)\}$$

The *ann* weighting formula converts documents d_{10} and d_{11} as

$$d_{10.ann} = \{(t_1, 1), \ldots, (t_m, 1), (t_{m+1}, 0), \ldots, (t_n, 0)\}$$

$$d_{11.ann} = \{(t_1, 1), \ldots, (t_m, 1), (t_{m+1}, 1), \ldots, (t_n, 1)\}$$

For query $q_3 = \{(t_1, w_1), \ldots, (t_m, w_m), (t_{m+1}, 0), \ldots, (t_n, 0)\}$, documents d_{10} and d_{11} are given the same similarity values as follows:

$$Sim(d_{4.lnc}, q_2) = Sim(d_{5.lnc}, q_2) = \sum_{i=1}^{m} w_i$$

Table 5: Number of common documents retrieved by the two runs using different weighting schemes (WSJ.D2: top-ranked 200 documents are retrieved for 100 queries)

	$lnc \cdot ltc$ (C)	$anc \cdot ltc$ (C)	$lnn \cdot ntc$ (N)	$ltn \cdot ntc$ (N)	$ann \cdot ntc$ (M)
$anc \cdot ltc$ (C)	**15183**	·	·	·	·
$lnn \cdot ntc$ (N)	9911	8200	·	·	·
$ltn \cdot ntc$ (N)	10573	9032	**15937**	·	·
$ann \cdot ntc$ (M)	10106	10395	13069	13310	·
$atn \cdot ntc$ (M)	10301	10627	11745	13733	**16069**

Even though this is not a general case, this example shows that the problem of cosine normalization may be alleviated by maximum normalization.

In this section we have addressed important properties of document weighting schemes, i.e. cosine normalization and maximum normalization to affect the document types retrieved. We classify weighting schemes into three classes depending on their properties as follows:

- *Class C* : Weighting schemes of class C perform cosine normalization. They may well retrieve single topic documents of being relevant in the collections with widely varying tf-vector length. However, cosine normalization may make it difficult to retrieve multiple topic documents of being relevant.

- *Class M* : Weighting schemes of class M perform maximum normalization, but do not perform cosine normalization. They cannot normalize tf-vector length in many cases that the weighting schemes of class C can. However, maximum normalization may alleviate the problem of cosine normalization on multiple topic documents.

- *Class N* : Weighting schemes of class N do not perform either cosine normalization or maximum normalization. Since the weighting schemes of this class do not have any normalization factor for tf-vector length, the documents of long tf-vector length are favored over those of short tf-vector length.

The above summary statements suggest that *different classes of weighting schemes may retrieve different types of documents - different sets of document (both relevant and nonrelevant).* We investigated the number of common documents retrieved by the two runs using different weighting schemes. We selected two weighting schemes for each class, and generated their pairwise combinations. Table 5 shows that more common documents are retrieved by the weighting schemes of the same class. The table also shows that the combinations between class C and the other classes have less common documents than those between class M and class N, which means that cosine normalization is a more important factor than maximum normalization in retrieving different sets of documents.

4 Combining Multiple Evidence

We have explained that different classes of weighting schemes may retrieve different types of documents - different sets of documents. In this section we perform a variety of retrieval runs using different weighting schemes with the WSJ.D2 collection, and combine the results. Both individual runs and combined runs retrieve top-ranked 200 documents in decreasing order of similarity values, and the 11-point average precision is used for their retrieval performance. First of all, we give the description of the combining method used in the experiments, and then present experimental results about the combination of individual runs.

4.1 Combining method

Since the retrieval runs using different weighting schemes generate quite different ranges of similarity values, a normalization method should be applied to each retrieval result. We normalize each similarity value by the maximum similarity value in a retrieval result, which will be called Max_Norm.

$$Max = \frac{old\_similarity}{maximum\_similarity}$$

Basically, normalization plays a role of controlling the ranges of similarities that retrieval systems generate. Max_Norm coincides only the upper bound of similarities. Hence, in order to coincide the lower bound as well as the upper, the following Min_Max_Norm looks more reasonable than Max_Norm.

$$Min\_Max = \frac{old\_similarity - minimum\_similarity}{maximum\_similarity - minimum\_similarity}$$

The minimum similarity generated by SMART is zero, in that SMART gives zero to the documents that do not have terms specified in a query. Therefore, Min_Max_Norm can be reduced to Max_Norm for SMART.

Since different runs have different levels of retrieval effectiveness, in general, it may be desirable or necessary to weight individual runs depending on their overall performance [14]. For example, suppose that two retrieval runs $r1$ and $r2$ give the 11-point average precision 0.2 and 0.3, respectively. If similarity values of $r1$ and $r2$ are multiplied by 0.2 and 0.3, respectively, a combined run may provide better retrieval effectiveness. However, we will not specially weight each of separate runs, and not favor any individual run. This is because the effectiveness of retrieval runs greatly depends on the characteristics of data collections, and it is difficult to estimate the performance of individual runs in an ad-hoc situation.

Fox and Shaw [5] have tested several functions of combining similarity values. As a result, the summation function, which sums up the set of similarity values, or, equivalently, the numerical mean of the set of similarity values works better in most TREC subcollections. In this paper we will also use the summation function for the combination of retrieval results as follows:

$$combined\_similarity = SUM(individual\_similarities)$$

Table 6: Combining the two runs using different weighting schemes

	$lnc \cdot ltc$ (C) 0.3284	$anc \cdot ltc$ (C) 0.3034	$lnn \cdot ntc$ (N) 0.2661	$ltn \cdot ntc$ (N) 0.3028	$ann \cdot ntc$ (M) 0.3002
$anc \cdot ltc$ (C) 0.3034	0.3205 (-2.4%)	·	·	·	·
$lnn \cdot ntc$ (N) 0.2661	0.3378 (+2.9%)	**0.3433 (+13.1%)**	·	·	·
$ltn \cdot ntc$ (N) 0.3028	**0.3464 (+5.5%)**	**0.3517 (+15.9%)**	0.2887 (-4.7%)	·	·
$ann \cdot ntc$ (M) 0.3002	**0.3575 (+8.9%)**	**0.3473 (+14.5%)**	0.2982 (-0.7%)	0.3162 (+4.4%)	·
$atn \cdot ntc$ (M) 0.3198	**0.3627 (+10.4%)**	**0.3451 (+7.9%)**	0.3130 (-2.2%)	0.3217 (+0.6%)	0.3148 (-1.6%)

Table 7: Combining the $lnc \cdot ltc$ run with the runs that perform cosine normalization

	$lnc \cdot ltc$	$lnc \cdot ltc$ $lnc \cdot ntc$	$lnc \cdot ltc$ $lnc \cdot atc$	$lnc \cdot ltc$ $anc \cdot ltc$	$lnc \cdot ltc$ $anc \cdot atc$	$lnc \cdot ltc$ $anc \cdot ntc$
Average Precision	0.3284	0.3202	0.3138	0.3205	0.3253	0.3236
% Change		-2.5	-4.5	-2.4	-0.9	-1.5

Table 8: Combining the $atn \cdot ntc$ run with the runs that do not perform cosine normalization

	$atn \cdot ntc$	$atn \cdot ntc$ $ltn \cdot ntc$	$atn \cdot ntc$ $ann \cdot ntc$	$atn \cdot ntc$ $atn \cdot ltc$	$atn \cdot ntc$ $ltn \cdot ltc$	$atn \cdot ntc$ $lnn \cdot ntc$
Average Precision	0.3198	0.3217	0.3148	0.3068	0.3160	0.3130
% Change		+0.6	-1.6	-4.1	-1.2	-2.2

Table 9: Combining the $lnc \cdot ltc$ run with the runs that do not perform cosine normalization

	$lnc \cdot ltc$	$lnc \cdot ltc$ $atn \cdot ntc$	$lnc \cdot ltc$ $ltn \cdot ntc$	$lnc \cdot ltc$ $ann \cdot ntc$	$lnc \cdot ltc$ $atn \cdot ltc$	$lnc \cdot ltc$ $ltn \cdot ltc$	$lnc \cdot ltc$ $lnn \cdot ntc$
Average Precision	0.3284	0.3627	0.3464	0.3575	0.3467	0.3378	0.3378
% Change		+10.4	+5.5	+8.9	+5.6	+2.9	+2.9

Table 10: Combining the $atn \cdot ntc$ run with the runs that perform cosine normalization

	$atn \cdot ntc$	$atn \cdot ntc$ $lnc \cdot ltc$	$atn \cdot ntc$ $lnc \cdot ntc$	$atn \cdot ntc$ $lnc \cdot atc$	$atn \cdot ntc$ $anc \cdot ltc$	$atn \cdot ntc$ $anc \cdot atc$	$atn \cdot ntc$ $anc \cdot ntc$
Average Precision	0.3198	0.3627	0.3540	0.3526	0.3450	0.3402	0.3451
% Change		+13.4	+10.7	+10.3	+7.9	+6.4	+7.9

4.2 Experiments

We applied the combining method to pairwise combinations of the six runs using different weighting schemes. These six runs were also used in Table 5 to show the number of common documents for the pairwise combinations. Performance results of the combined runs are presented in Table 6, in which % change is given with respect to the run providing better effectiveness in each combination. The results in Table 5 have some coincidence with those in Table 6, in that *the more different documents two different runs retrieve, the more improvements their combination results in if there is no much difference between the effectiveness of the two runs.* When we combine $ann \cdot ntc$ and $anc \cdot ltc$, the number of common documents is 10395, and the improvement is +14.5%. However, at the combination of $lnn \cdot ntc$ and $lnc \cdot ltc$, we get only +2.9% improvement even though the number of common retrieved items, i.e. 9911 is a bit less than 10395. It should be noticed that the effectiveness of combined runs is affected by the effectiveness of individual runs as well as the number of common documents. In combining $lnn \cdot ntc$ and $lnc \cdot ltc$, the small improvement may be due to the fact that the effectiveness of $lnn \cdot ntc$, i.e. 0.2661 is much lower than that of $lnc \cdot ltc$, i.e. 0.3284.

Table 6 shows that significant improvements can be obtained only for the combinations between the weighting schemes of class C and those of other classes, i.e. class M and class N. In other words, *we can get significant improvements by combining the two runs in which one performs cosine normalization and the other does not if the two runs provide similar level of retrieval effectiveness.* More experimental evidence is given in the remainder of this section in order to confirm this result.

We have selected six of the class C weighting schemes, i.e. top-effective six weighting schemes in Table 2, and six of the others, i.e. top-effective six weighting schemes in Table 3. Then, we generated some pairwise combinations and applied the combining method. Table 7 and 8 show that we cannot get any improvement by combining the two runs that perform cosine normalization, or by combining the two runs that do not perform cosine normalization. Table 9 and 10 show that significant improvements can be achieved by combining the two runs in which one performs cosine normalization and the other does not.

We performed the p-norm extended Boolean retrieval run [15, 16] in which document terms are weighted with ann. Then, we combined the results of the p-norm run with those of two vector runs such as $lnc \cdot ltc$ and $atn \cdot ntc$. It is known that the p-norm extended Boolean model has desirable properties providing high retrieval effectiveness than any other extended Boolean models. The p-norm extended Boolean model can calculate the similarities of documents with respect to p-norm type queries that are composed of terms, logical operators such as AND, OR and NOT, p-values for logical operators, and weights for terms and clauses. We got conventional Boolean queries formulated from topic descriptions of the TREC data collections at Virginia Tech [5], and created p-norm type queries by using the uniform p-value of 1.5 and calculating the weights of terms and clauses with the normalized idf weight and the sum weight [17], respectively. Table 11 shows that combining p-norm and $lnc \cdot ltc$ provides better improvement than combining p-norm and $atn \cdot ntc$ even though $lnc \cdot ltc$ and $atn \cdot ntc$ have the similar level of effectiveness. It may be due to the fact that p-norm and $lnc \cdot ltc$ use different classes of weighting schemes whereas p-norm and $atn \cdot ntc$ use the same classes. Note that the document weighting scheme of $lnc \cdot ltc$ has the

Table 11: Combining the p-norm run with vector runs

	$lnc \cdot ltc$	$atn \cdot ntc$	p-norm	$lnc \cdot ltc$ p-norm	$atn \cdot ntc$ p-norm
Average Precision	0.3284	0.3198	0.3173	0.3744	0.3454
% Change				+14.0% over $lnc \cdot ltc$	+8.0% over $atn \cdot ntc$

Table 12: Combining the $lnc \cdot ltc$ and $atn \cdot ntc$ runs

		$lnc \cdot ltc$	$atn \cdot ntc$	$lnc \cdot ltc$ $atn \cdot ntc$
AP.D2	Average Precision	0.3878	0.3783	0.4131
	% Change			+6.5% over $lnc \cdot ltc$
ZIFF.D2	Average Precision	0.1897	0.1687	0.2198
	% Change			+15.9% over $lnc \cdot ltc$
FR.D2	Average Precision	0.1187	0.1273	0.1500
	% Change			+17.8% over $atn \cdot ntc$
WSJ.D2	Average Precision	0.3284	0.3198	0.3627
	% Change			+10.4% over $lnc \cdot ltc$

cosine normalization factor, and those of p-norm and $atn \cdot ntc$ do not.

Finally, we evaluated $lnc \cdot ltc$ and $atn \cdot ntc$ with other TREC subcollections called AP newswire Disk 2 (AP.D2), ZIFF-davis publishing Disk 2 (ZIFF.D2) and Federal Register Disk 2 (FR.D2) [11]. Note that $lnc \cdot ltc$ is the most effective in the weighting schemes performing cosine normalization, and $atn \cdot ntc$ in the others. Table 12 shows that combining the $lnc \cdot ltc$ and $atn \cdot ntc$ runs gives significant improvements for various data collections.

5 Conclusion

Various strategies for representing queries and documents, and various retrieval techniques are available these days in the information retrieval literature. Several researchers have investigated the effect of combining multiple representations of either queries or documents, or multiple retrieval techniques on retrieval performance because different representations or different retrieval techniques can retrieve different documents. Recent work has shown that significant improvements can be achieved by the combination of multiple evidence.

In this paper we have proposed a method for retrieving different document sets. The method can be easily incorporated in the system using a single query representation, a single document representation and a single retrieval technique. We have classified the types of documents depending on tf-vector length and the number of topics described in the documents, and have described the properties of weighting schemes such as cosine normalization and maximum normalization. Then, we have explained that different types of documents may be retrieved by different properties of

weighting schemes. We have also shown through experiments that significant improvements can be obtained by combining the two retrieval runs in which one performs cosine normalization and the other does not.

Information retrieval systems, which can calculate query-document similarities, normally index queries and documents with only a single weighting scheme. The results described in this paper suggest that using different properties of two or more weighting schemes should provide better retrieval performance than using only one weighting scheme. However, we investigated only the weighting schemes of the SMART system. A variety of weighting schemes developed in the area of information retrieval should be analyzed to get more general properties of weighting schemes.

Acknowledgments

I would like to thank Gerard Salton for reading earlier drafts of this paper. His comments have greatly improved the quality of this paper. I would also like to give special thanks to Chris Buckley for his help in setting up experimental environment.

References

[1] M. McGill, M. Koll and T. Norreault, "An evaluation of factors affecting document ranking by information retrieval systems," Syracuse, Syracuse University School of Information Studies, 1979.

[2] J. Katzer, M.J. McGill, J.A. Tessier, W. Frakes and P. Dasgupta, "A study of the overlap among document representations," Information Technology: Research and Development, Vol. 1, No. 2, pp. 261-274, 1982.

[3] T. Saracevic and P. Kantor, "A study of information seeking and retrieving. III. Searchers, searches, overlap," Journal of the American Society for Information Science, Vol. 39, No. 3, pp. 197-216, 1988.

[4] H. Turtle and W.B. Croft, "Evaluation of an inference network-based retrieval model," ACM Transactions on Information Systems, Vol. 9, No. 3, pp. 187-222, 1991.

[5] E.A. Fox and J.A. Shaw, "Combination of multiple searches," Proceedings of the 2nd Text REtrieval Conference (TREC-2), National Institute of Standards and Technology Special Publication 500-215, pp. 243-252, 1994.

[6] N.J. Belkin, C. Cool, W.B. Croft and J.P. Callan, "The effect of multiple query representations on information retrieval performance," Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 339-346, 1993.

[7] D. Harman, "Overview of the 1st text retrieval conference," Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.36-48, 1993.

[8] G. Salton and M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, Inc., 1983.

[9] G. Salton, Automatic Text Processing - the Transformation, Analysis and Retrieval of Information by Computer, Addison-Wesley Publishing Co., Reading, MA, 1989.

[10] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Information Processing and Management, Vol. 24, No. 5, pp. 513-523, 1988.

[11] D. Harman, "Overview of the second text retrieval conference," Proceedings of the 2nd Text REtrieval Conference (TREC-2), National Institute of Standards and Technology Special Publication 500-215, pp. 243-252, 1994.

[12] A. Moffat, R. Sacks-Davis, R. Wilkinson, and J. Zobel, "Retrieval of partial documents," Proceedings of the Second Text REtrieval Conference (TREC-2). National Institute of Standards and Technology Special Publication 500-215, pp. 181-190, 1994.

[13] J.P. Callan, "Passage-level evidence in document retrieval," Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 302-310, 1994.

[14] B.T. Bartell, G.W. Cottrell and R.K. Belew, "Automatic Combination of Multiple Ranked Retrieval Systems," Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 173-181, 1994.

[15] J.H. Lee, Y.W. Kim, M.H. Kim and Y.J. Lee, "On the evaluation of Boolean operators in the extended Boolean retrieval framework," Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 291-297, 1993.

[16] J.H. Lee, "Properties of extended Boolean models in information retrieval," Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 182-190, 1994.

[17] M.E. Smith, Aspects of the P-norm Model of Information Retrieval: Syntactic Query Generation, Efficiency, and Theoretical Properties, Ph.D. Thesis, Department of Computer Science, Cornell University, 1990.