Pivoted Document Length Normalization

Amit Singhal*, Chris Buckley, Mandar Mitra, (Gerard Salton)

Department of Computer Science, Cornell University, Ithaca, NY 14853
{singhal, chrisb, mitra}@cs.cornell.edu

Abstract

Document length normalization is an important aspect of term weight assignment in an automatic information retrieval system. In this study, we observe that a normalization scheme that retrieves documents of all lengths with similar chances as their likelihood of relevance will outperform another scheme which retrieves documents with chances very different from their likelihood of relevance. We show that the retrieval probabilities for a particular normalization method deviate systematically from the relevance probabilities across different collections. We present *pivoted normalization*, a technique that can be used to reduce the gap between the relevance and the retrieval probabilities. Training pivoted normalization on one collection, we can successfully use it on other (new) text collections, yielding a robust, *collection independent* normalization technique. We use the idea of pivoting with the well known cosine normalization scheme. We point out some shortcomings of the cosine normalization function and present two new normalization functions — *pivoted unique normalization* and *pivoted byte size normalization*.

1 Background

Term weighting is an important aspect of modern text retrieval systems. [3] Terms are words, phrases, or any other indexing units used to identify the contents of a text. Since different terms have different importance in a text, an importance indicator — the *term weight* — is associated with every term. [11] Three main components that affect the importance of a term in a text are the term frequency factor (*tf*), the inverse document frequency factor (*idf*), and document length normalization. [12]

Term weight normalization is used to remove the advantage that the long documents have in retrieval over the short documents. Two main reasons that necessitate the use of normalization in term weights are:

1. **Higher term frequencies:** Long documents usually use the same terms repeatedly. As a result, the term frequency factors may be large for long documents, increasing the average contribution of its terms towards the query–document similarity.
2. **More terms:** Long documents also have numerous different terms. This increases the number of matches between a query and a long document, increasing the query–document similarity, and the chances of retrieval of long documents in preference over shorter documents.

Normalization is a way of penalizing the term weights for a document in accordance with its length. Various normalization techniques are used in information retrieval systems. Following is a review of some commonly used normalization techniques:

- **Cosine:** Cosine normalization is the most commonly used normalization technique in the vector space model. [13] The cosine normalization factor is computed as $\sqrt{{w_1}^2 + {w_2}^2 + \ldots + {w_t}^2}$, where w_i is the raw $tf \times idf$ weight for a term. [9, 11] Cosine normalization attacks both the reasons for normalization (*higher tfs* and *more terms*) in one step. Higher individual term frequencies increase individual w_i values, increasing the penalty on the term weights. Also, if a document has more terms, the number of individual weights in the cosine factor (t in the above formula) increases, yielding a higher normalization factor.

*This study was supported in part by the National Science Foundation under grant IRI-9300124.

- **Maximum tf:** Another popular normalization technique is normalization of individual *tf* weights for a document by the maximum *tf* in the document. The Smart system's augmented *tf* factor ($0.5 + 0.5 \times \frac{tf}{max\_tf}$), and the *tf* weights used in the INQUERY system ($0.4 + 0.6 \times \frac{tf}{max\_tf}$) are examples of such normalization. [11, 16] By restricting the *tf* factors to a maximum value of 1.0, this technique only compensates for the first reason (*higher tfs*) for normalization. When used without any correction for the second reason (*more terms*) this turns out to be a "weak" form of normalization and favors the retrieval of long documents. [2]

- **Byte Length:** More recently, a length normalization scheme based on the byte size of documents has been used in the Okapi system. [8] This normalization factor attacks both the reasons for normalization in one shot.

This study shows that better retrieval effectiveness results when a retrieval strategy retrieves documents with chances similar to their probability of relevance. We present a technique to analyze these probabilities. Based on observations from this analysis, we present a novel normalization approach — *pivoted normalization*. We show that pivoted normalization yields substantial improvements in retrieval effectiveness.

The rest of this study is organized as follows. Section two introduces pivoted normalization. Section three shows how the cosine function can be pivoted to obtain significant improvements in retrieval effectiveness. Section four further analyzes the cosine function. Section five introduces pivoted unique normalization, another possible function for document length normalization. Section six introduces pivoted byte size normalization for use in degraded text collections. Section seven concludes the study.

2 Approach

For a document collection and a set of queries, we analyze the likelihood of relevance/retrieval for documents of all lengths, and plot these likelihoods against the document length to obtain a "relevance pattern" and a "retrieval pattern". In general, a scheme under which the probability of retrieval for the documents of a given length is very close to the probability of finding a relevant document of that length should perform better than another scheme which retrieves documents with very different chances from their relevance probability. The aim is, then, to learn how the retrieval pattern deviates from the relevance pattern for a given normalization function. Under the hypothesis that this deviation is systematic across different queries and different document collections, we can propose *collection independent* techniques to reduce this deviation.

2.1 Likelihood of Relevance/Retrieval

To design functions that attempt to match the likelihood of retrieval to the likelihood of relevance, we need a way to estimate these likelihoods. We do this by ordering the documents in a collection by their lengths, and dividing them into several equal sized "bins". We can then compute the probability of a randomly selected relevant/retrieved document belonging to a certain bin. For example, to do such an analysis for fifty TREC [6] queries (151–200) and 741,856 TREC documents (from disks one and two), we sorted the documents in order of increasing byte-length. We divided this sorted list into bins of one thousand documents each, yielding 742 different bins: the first 741 bins containing one thousand documents each, and the last bin containing the longest 856 documents. We selected the *median* document length in each bin to represent the bin on the graphs used in later analysis.

We took the 9,805 ⟨query, relevant-document⟩ pairs for the fifty queries, and counted how many pairs had their document from the ith bin. We then computed the probability that a randomly selected relevant document belongs to the ith bin — the ratio of the number of pairs that have their document from the ith bin, and the total number of pairs (9,805). In terms of conditional probability, given a document D, this ratio for the ith bin can be represented by $P(D \in Bin_i \mid D \textit{ is Relevant})$. Similarly, by retrieving the top one thousand documents for each query (yielding 50,000 ⟨query, retrieved-document⟩ pairs), and repeating the above analysis for a bin, we get the conditional probability of retrieval, $P(D \in Bin_i \mid D \textit{ is Retrieved})$, for a particular normalization function.

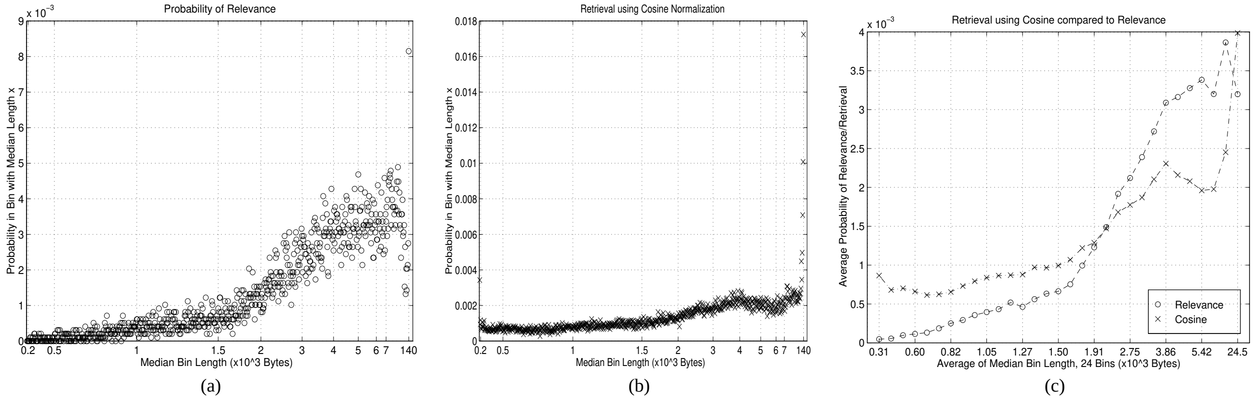


Figure 1: Probability that a relevant/retrieved document is from a bin, plotted against the median bin length. The analysis for the relevant documents is shown in (a), (b) shows the analysis for documents retrieved using cosine normalization, and (c) compares the smooth plots for (a) and (b).

Figures 1(a) and 1(b) show the plots of the probabilities obtained from the above analysis plotted against the median document length in a bin. Smart's *lnc.ltc* retrieval, which is based upon cosine normalization, was used to get the retrieval probabilities. [4] In Figure 1(c), the smoothed plots[1] for the relevance and the retrieval probabilities are graphed together. This comparison reveals important information about the length normalization properties of a term weighting strategy. For example, we can observe from the smoothed plots that *lnc.ltc* retrieval has a tendency to retrieve short documents with a higher probability than their probability of relevance; it is less likely to retrieve longer documents as compared to the likelihood of their relevance. This observation reinforces the long held belief that *cosine normalization tends to favor short documents in retrieval*. When using *lnc.ltc* retrieval, we would like to (somehow) promote the retrieval of longer documents, and we would like to retrieve fewer short documents.

2.2 The "Pivoted" Normalization Scheme

The higher the value of the normalization factor for a document is, the lower are the chances of retrieval for that document. In effect, the probability of retrieval of a document is inversely related to the normalization factor used in the term weight estimation for that document. This relationship suggests that to boost the chances of retrieval for documents of a certain length, we should lower the value of the normalization factor for those documents, and vice-versa. The pivoted normalization scheme is based on this principle.

The basic idea of pivoted normalization is illustrated in Figure 2. Using a normalization function (like cosine, or byte-size), a set of documents is initially retrieved. As shown in Figure 1(c), the retrieval and the relevance curves are plotted. The point where these two curves cross each other is called the *pivot*. The documents on one side of the pivot are generally retrieved with a higher probability than their relevance probability, and the documents on the other side of the pivot are retrieved with a lower probability than their probability of relevance. The normalization function can now be "pivoted" at the pivot and "tilted" to increase the value of the normalization factor, as compared to the original normalization factor, on one side of the pivot. This also decreases the value of the normalization factor on the other side of the pivot. The amount of "tilting" needed becomes a parameter of the weighting scheme, and is called the *slope*. With such pivoting and tilting, the pivoted normalization factor is represented by the equation for a line of gradient *slope* that intersects the line of unit gradient at the point *pivot*.

$$pivoted\ normalization = (1.0 - slope) \times pivot + slope \times old\ normalization \tag{1}$$

If this deviation of the retrieval pattern from the relevance pattern is systematic across collections for a

[1] We generated smooth plots for various figures by representing a sequence of 24 bins by a single point and connecting these points by a curve. The 742 bins yielded 31 different points where the last point represented the longest 22 bins ($742 = 30 \times 24 + 1 \times 22$). The representative point for a group of bins was obtained by taking the averages of both the median lengths, and the probabilities of relevance/retrieval for the 24 consecutive bins.

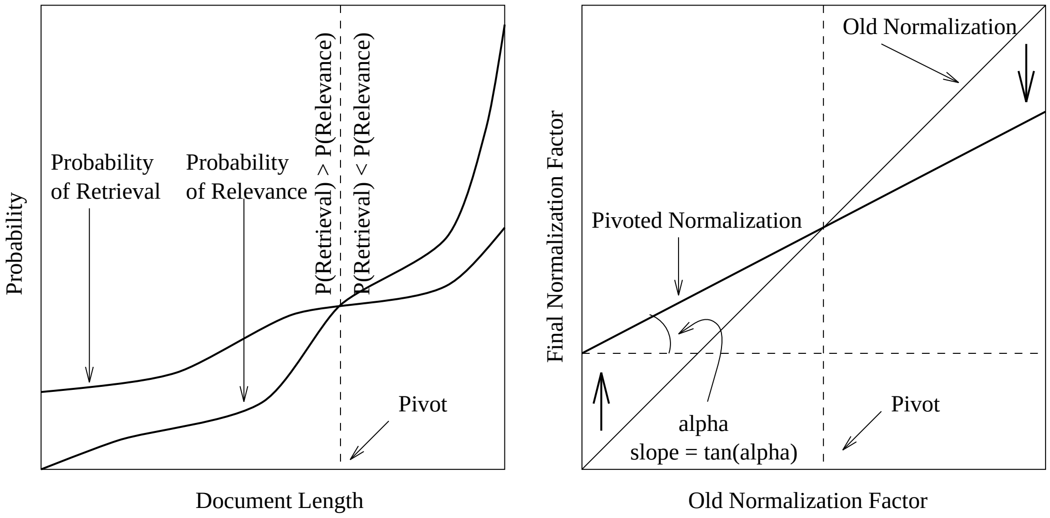


Figure 2: **Pivoted Normalization.** The normalization factor for documents for which *P(retrieval)* $>$ *P(relevance)* is increased, whereas the normalization factor for documents for which *P(retrieval)* $<$ *P(relevance)* is decreased.

normalization function, the pivot and the slope values learned from one collection can be used effectively on another collection. See [15] for a more detailed description of this technique.

2.3 Removing One Parameter

Using pivoted normalization, the new weight of a document term can be written as:

$$\frac{tf \cdot idf \;\; weight}{(1.0 - slope) \times pivot + slope \times old \;\; normalization}$$

If we multiply every document term weight by a constant, the relative ranking of the documents under inner-product similarity measurement remains unchanged since individual document similarities are simply scaled by the constant. Multiplying each weight by the constant $(1.0 - slope) \times pivot$, we obtain the following term weighting formula:

$$\frac{tf \cdot idf \;\; weight \times (1.0 - slope) \times pivot}{(1.0 - slope) \times pivot + slope \times old \;\; normalization} \quad or \quad \frac{tf \cdot idf \;\; weight}{1 + \frac{slope}{(1.0 - slope) \times pivot} \times old \;\; normalization}$$

We observe that the form of the pivoted normalization function is $1 + c \times old \;\; normalization$, where the constant c equals $\frac{slope}{(1.0 - slope) \times pivot}$. If the pivot value in an optimal constant c is changed to $pivot'$, the slope value can be modified to $slope'$ to get back the optimal constant. If we fix the pivot value at some collection specific value, like the *average old normalization factor*, it is still possible to obtain an optimal slope value by training. Therefore, the number of parameters that we need to train for is reduced to just one instead of two.

Selecting the *average old normalization factor* as the pivot has a nice interpretation. If instead of multiplying every term weight by $(1.0 - slope) \times pivot$ in Equation 1, we multiply every weight by the constant *pivot* (which has the value *average old normalization*), the final normalization factor reduces to:

$$(1.0 - slope) + slope \times \frac{old \;\; normalization}{average \;\; old \;\; normalization}$$

From this expression, similar to Robertson's notion [7], we can say that an *average length document* is of "appropriate length" and its weights should remain unchanged, *i.e.*, it should get *unit* (or no) normalization. Also, the slope can be interpreted as our "belief in length".

Cosine	Pivoted Cosine Normalization				
	Slope				
	0.60	0.65	0.70	**0.75**	0.80
6,526	6,342	6,458	6,574	**6,629**	6,671
0.2840	0.3024	0.3097	0.3144	**0.3171**	0.3162
Improvement	+ 6.5%	+ 9.0%	+10.7%	**+11.7%**	+11.3%

Table 1: Estimation of a good slope in pivoted cosine normalization. The pivot is set to the average cosine normalization factor (13.36) for TREC disks one and two (741,856 documents). TREC queries 151–200 were used in these experiments. Each entry shows the total number of relevant documents retrieved (out of 9,805) for all fifty queries, the non-interpolated average precision, and the improvement in average precision over using cosine normalization.

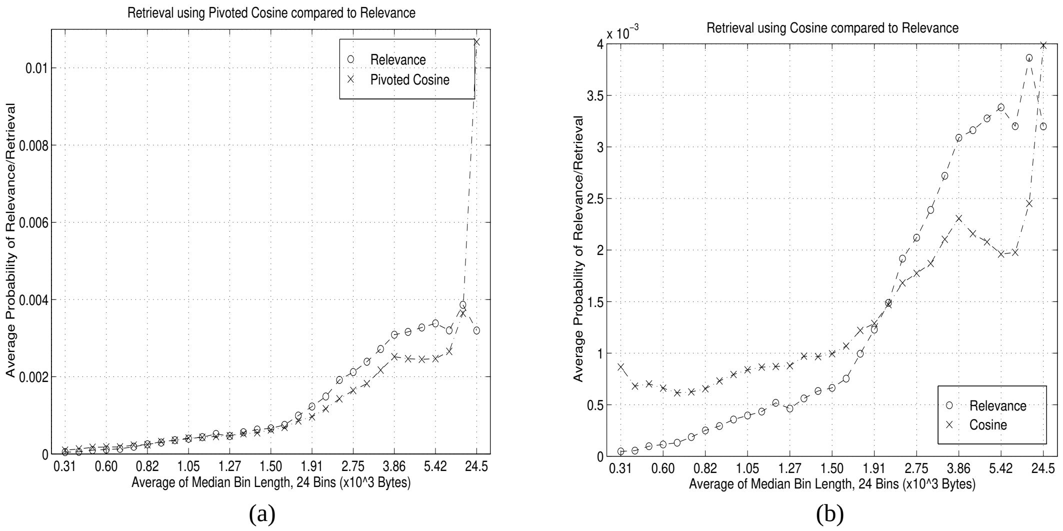


Figure 3: Pivoted cosine normalization: comparison of retrieval pattern relevance pattern (a), and same comparison for cosine normalization (b).

3 Pivoted Cosine Normalization

Since cosine normalization is the most commonly used normalization technique, it is natural to test pivoting with the cosine function first. In our studies with the TREC collection [6], a *tf* factor of $1 + log(tf)$ works well. Also, the *idf* factor is only used in the query term weights and not in the document term weights. [4] Fixing the pivot value at the average cosine factor for $1 + log(tf)$ weighted documents for TREC disks one and two (average = 13.36), we retrospectively learn the value of a good slope for TREC queries 151-200 (see Table 1). Substantial improvements over cosine normalization — 9–12% improvement in average precision — are obtained using pivoted cosine normalization.

Figure 3(a) compares the retrieval pattern for pivoted cosine normalization to the relevance pattern. For comparison with cosine normalization, Figure 1(c) has been reproduced here as Figure 3(b). We observe that the curve for the retrieval probability using pivoted cosine normalization is much closer to the curve for the relevance probability, as opposed to the curve for retrieval using cosine normalization. This indicates that pivoted cosine normalization retrieves documents of all lengths with chances much closer to their likelihood of relevance. This observation along with the **11.7%** improvement over cosine normalization strongly supports our hypothesis that schemes that retrieve documents of different lengths with chances similar to their likelihood of relevance will have a higher retrieval effectiveness. To test the robustness of pivoted cosine normalization, we tested it on the other 150 TREC queries (1–150). The training for slope for TREC queries 1–150 is shown in Table 2. Once again we see that pivoted cosine normalization yields 10–12% improvement over cosine normalization.

As relevance judgments are not available in an adhoc querying environment, to observe the variability in a good slope value across query sets, we also tested the optimal slope value obtained from a set of training

Cosine	Pivoted Cosine Normalization				
	Slope				
	0.60	0.65	**0.70**	0.75	0.80
28,484	30,270	30,389	**30,407**	30,314	30,119
0.3063	0.3405	0.3427	**0.3427**	0.3411	0.3375
Improvement	+11.2%	+11.9%	**+11.9%**	+11.4%	+10.2%

Table 2: Estimation of a good slope in pivoted cosine normalization for TREC queries 1–150. Each entry shows the total number of relevant documents retrieved (out of 46,555) for all 150 queries, the non-interpolated average precision, and the improvement in average precision over cosine normalization.

queries (TREC queries 1–150) on a set of test queries (TREC queries 151–200). We observe from Table 2 that the best slope value for queries 1–150 is 0.70. If we use this slope value for queries 151–200, we would still achieve "near best" performance — 10.7% improvement in place of 11.7% (see Table 1). This indicates that it is possible to learn the slope value on one set of queries and successfully use it on another.

To test our hypothesis that the deviation of the retrieval pattern from the relevance pattern for a given normalization function is systematic across different query sets and different document collections, we studied these patterns for cosine normalization on six different sub-collections of the TREC collection. [6] Figure 4 shows the relevance patterns and the retrieval patterns (for queries that have any relevant document in a collection) obtained using cosine normalization and pivoted cosine normalization for various collections. We observe that, despite the widely varying relevance patterns for different collections, for cosine normalization, the deviation of the retrieval pattern from the relevance pattern is indeed systematic. For all collections, use of cosine normalization retrieves short documents with chances higher than their likelihood of relevance, and retrieves long documents with chances lower than their likelihood of relevance. Using pivoted cosine normalization reduces the gap between the retrieval and the relevance pattern for all the collections. Moreover, the slope value learned from one collection is near optimal — within 5% of the best slope value — for all the collections. Using a slope of 0.70 across collections, important improvements (+4.3% to +21.7%) are achieved on all the collections.

4 Analysis of the Cosine Function

On close observation of Figure 1(b) we notice that when cosine normalization is used for TREC queries 151–200, the probability of retrieval for the last few bins, containing "extremely" long documents, is substantially higher than the rest of the collection. The last few bins contain documents that are longer than 20,000 bytes, more than six times the average document size for the entire collection. This favoring of extremely long is more prevalent when pivoted cosine normalization is used — the last few bins in Figure 3(a) have very high retrieval probabilities.

This favoring is further examined in Figure 5 which shows a magnified view of the long end of the document length spectrum, the last twenty bins. We notice that using cosine normalization, the retrieval probabilities for extremely long documents are marginally greater than their probability of relevance, *i.e.*, cosine normalization retrieves these documents with "mildly higher" chances than we would like. When we use pivoted cosine normalization, which aims at favoring long documents, we end up "strongly favoring" extremely long documents. This effect causes excessive retrieval of such (possibly non-relevant) documents, hurting the retrieval effectiveness.

On deeper analysis of the cosine function, we observe that if all the terms appear just once in a document ($tf = 1$), the cosine normalization factor for the document is (individual term weights are $1 + log(tf) = 1$ and we are not using the *idf* factor on documents):

$$\sqrt{1^2 + 1^2 + \ldots 1^2} = \sqrt{\#\ of\ \ unique\ \ terms}$$

In reality, some terms occur more than once in a document, and the cosine factor can be higher than $\sqrt{\#\ of\ \ unique\ \ terms}$. In practice, however, the cosine normalization factors for documents are very close to the function $\sqrt{\#\ of\ \ unique\ \ terms}$ due to the following two facts:

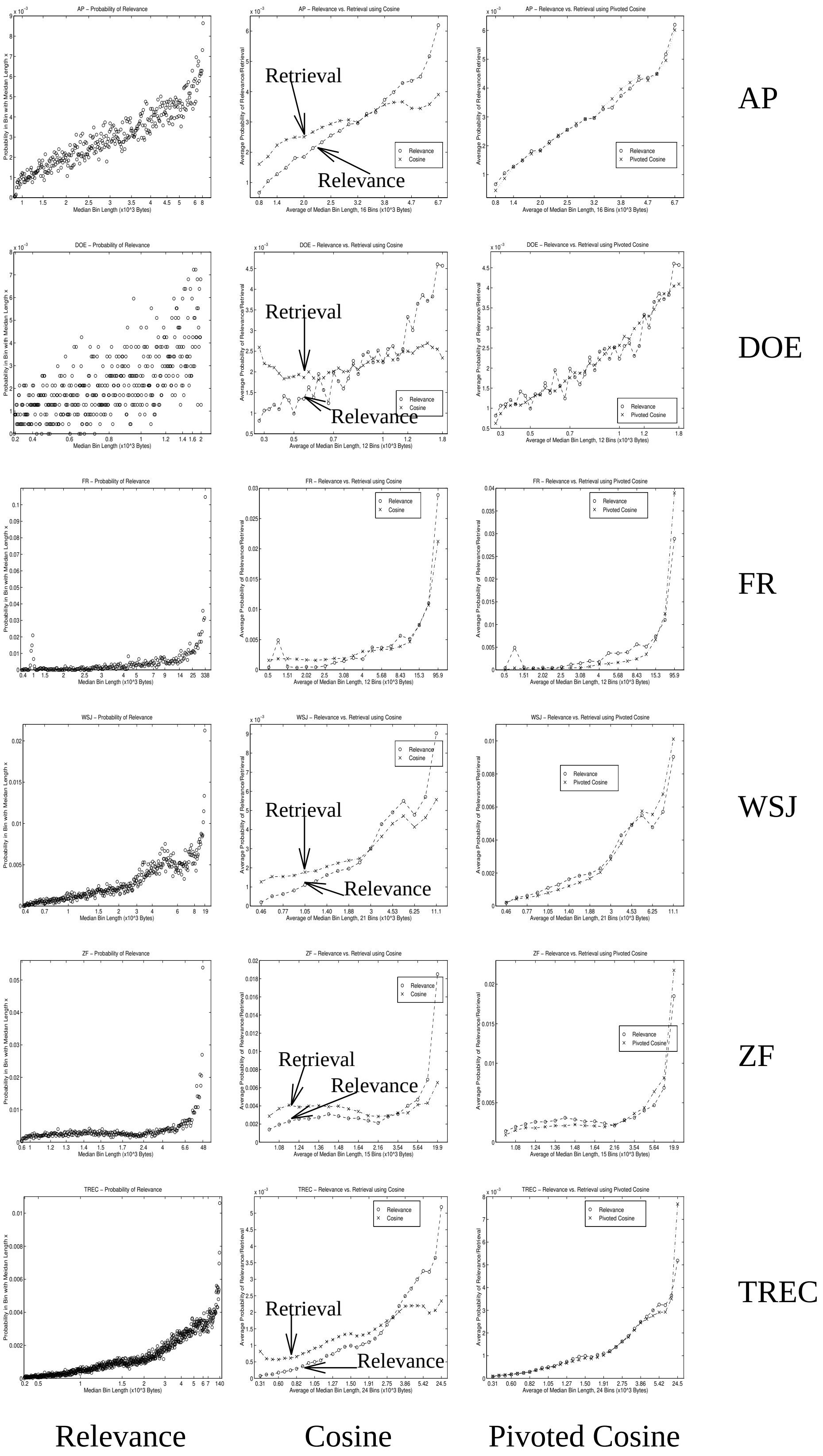


Figure 4: Comparison of cosine and pivoted cosine normalization for six different collections. Use of cosine normalization invariably favors short documents in retrieval. This problem is reduced by the use of pivoted cosine normalization.

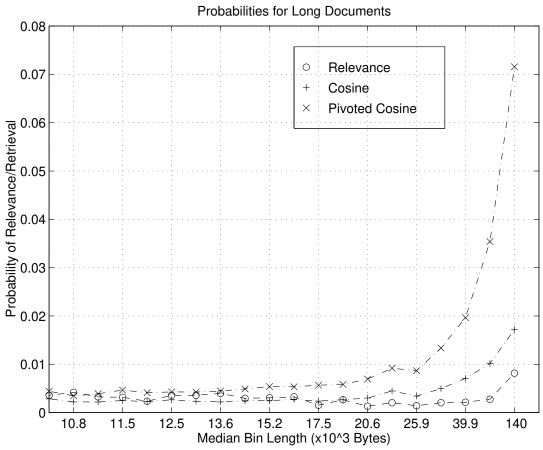


Figure 5: Probabilities in the last twenty bins containing the longest 19,856 documents (last bin has only 856 documents) from the collection. We notice that pivoted cosine normalization favors the retrieval of extremely long documents.

- It is well known that the majority of the terms in a document occur only once. So there are only a few terms that have $tf > 1$.
- As we use $1 + log(tf)$ as the tf factor, for most of the terms with $tf > 1$, the tf factors are "not too big". Due to the "dampening effect" of the *log* function, most of the *tf* factors, in practice, are "near one".

When we studied the variation of the cosine factor for TREC documents in relation to the number of unique terms in a document, we observed that the cosine factor actually does vary like the function $\#\ of\ unique\ terms^{0.6}$.

Further, with dampened *tf* factors, even with high raw *tf* values in individual documents, document retrieval is not strongly affected by the term frequency factors. The retrieval of documents is generally governed by the number of matches to the query. Assuming complete binary term independence, the probability of a match between the query and a document increases *linearly* in the number of different terms in a document[2]. Therefore a good length normalization function should also vary linearly with the number of unique terms in a document.

As documents grow longer, the cosine function, with its variation as $\#\ of\ unique\ terms^{0.6}$, becomes substantially weaker than a linear function in *# of unique terms*. For this reason, we observe that the use of cosine function can favor extremely long documents in retrieval. This problem is aggravated with the use of pivoted cosine normalization which further aids the retrieval of long documents. We propose that a function linear in the number of unique terms in a document be used for normalization.

5 Pivoted Unique Normalization

Based on the above observations, we use the number of unique terms in a document as the normalization function, and to match the likelihoods of relevance and retrieval, we use pivoting of this function to get the pivoted unique normalization function:

$$pivoted\ unique\ normalization = (1.0 - slope) \times pivot + slope \times \#\ of\ unique\ terms$$

Since the pivoted unique normalization factor only compensates for the second effect — *more terms* in long documents — that necessitates (the presence of) normalization, we still need to compensate for the first effect — *higher tfs* in long documents (see Section 1). Normalization of *tf* weights by maximum *tf*

[2] Suppose the vocabulary size is T, and document D has k different terms. The probability that a randomly selected query term belongs to document D is $\frac{k}{T}$. This probability increases linearly in k.

Cosine	Pivoted Unique Normalization			
	Slope			
	0.15	0.20	**0.25**	0.30
6,526 0.2840	6,688 0.3268	6,841 0.3355	**6,864** **0.3361**	6,852 0.3318
Improvement	+15.1%	+18.1%	**+18.3%**	+16.8%
Improvement over best (0.3171) Pivoted Cosine	+ 3.1%	+ 5.8%	**+ 6.0%**	+ 4.6%

Table 3: Estimation of a good slope in pivoted unique normalization for TREC queries 151–200. Each entry shows the total number of relevant documents retrieved (out of 9,805) for all fifty queries, and the non-interpolated average precision. The improvements in average precision over cosine normalization and over pivoted cosine normalization are also shown. The pivot value was set to 107.89, which is the average number of unique terms in a document for TREC disks one and two.

in a document can possibly be used to remove this effect, but we believe that *max_tf* is not an optimal normalization scheme to fix the *higher tfs* problem. For example, if a query term occurs five times in document D_1 in which all other terms occur just once, then D_1 is possibly more interesting than another document D_2 in which the same query term occurs five times as well, but all other terms also occur five times each. If *max_tf* is used for normalization, D_1 has no advantage over D_2 since the query term will have the same weight in both the documents.

We believe that *average term frequency* in a document is a better representative of the "verbosity" of a document. Judging term importance by term frequencies, if all terms were equally important in a document, each should occur the same number of times in that document with $tf = average\ \ tf$. For this reason, we would like a term that has $tf = average\ \ tf$ to have *unit importance* in a document. We use the function:

$$\frac{1 + log(tf)}{1 + log(average\ \ tf)}$$

as the term frequency factor for a term in a document. In experiments comparing *average term frequency* based normalization to *maximum term frequency* based normalization[3] (in conjunction with pivoted unique normalization with retrospectively trained slope value), we observed that average term frequency based normalization performed 5.7% better for 200 TREC queries on the entire TREC collection.

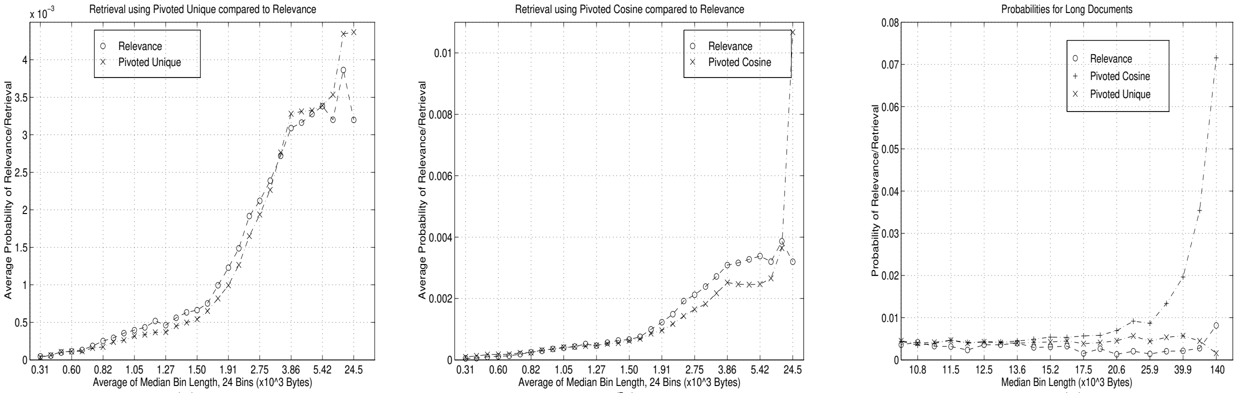
Based on this *tf* factor (which we call the L factor in Smart's term weight triple notation [11]) and pivoted unique normalization (which we call the u normalization factor), we obtain the final weighting strategy of the documents (called *Lnu* weighting in Smart):

$$\frac{\frac{1+log(tf)}{1+log(average\ \ tf)}}{(1.0 - slope) \times pivot + slope \times \#\ of\ \ unique\ \ terms}$$

Once again, we can use the average number of unique terms in a document (computed across the entire collection) as the pivot, and train for a good slope value.

The results of switching to pivoted unique normalization from pivoted cosine normalization for TREC queries 151–200 are listed in Table 3. We observe that the best pivoted unique normalization yields another 6% improvement over the best pivoted cosine normalization, resulting in an overall **18.3%** improvement over cosine normalization. A deeper analysis of retrieval using *Lnu* weighted documents (Figure 6(a)) reveals that in comparison to pivoted cosine normalization (Figure 6(b)), the probability of retrieval using pivoted unique normalization is, in fact, even closer to the probability of relevance for documents of all lengths. We also notice in Figure 6(c) that the advantage that very long documents had by the use of pivoted cosine normalization is removed by using pivoted unique normalization. The additional 6% improvement in Table 3 shows that as the retrieval probabilities come closer to the relevance probabilities, retrieval effectiveness increases. The closer the two curves are, the higher is the retrieval effectiveness.

[3] We used the function $0.4 + 0.6 \times \frac{1+log(tf)}{1+log(max\_tf)}$, a function similar to the well tested and effective function of the INQUERY system. [2]



(a) (b) (c)

Figure 6: Pivoted unique normalization compared to pivoted cosine normalization.

To verify the general applicability of pivoted unique normalization schemes, we also tested it on various sub-collections of TREC. Substantial improvements over cosine normalization are obtained for all the collections. Also, the slope value is very stable, *i.e.*, the changes in retrieval effectiveness with minor deviations in slope (from the optimal slope value) are very small for all the collections. A constant slope value of 0.20 was effective across collections. These observations are reassuring in terms of the general applicability of the pivoted normalization schemes.

6 Degraded Text Collections

When large text collections are constructed by electronically scanning the documents and using optical character recognition (OCR), the resulting text is usually degraded because of faulty recognition by the OCR process. Term weighting strategies that are effective for correct text collections might not be effective for degraded text collections. For example, if we use pivoted unique normalization in a degraded text collection, the normalization factor for documents will be affected by the poor quality of the input text (usually the number of unique terms in a document will be artificially high because different occurrences of a term can yield different unique terms in the degraded text).

Term weighting strategies that are not affected by the errors in the input text are needed for degraded text collections. [14] For correct collections, we have used the cosine factor and the number of unique terms to represent a document's length. In a degraded text collection, length measures that undergo little distortion in the OCR process should be used for document length normalization. Since longer documents have more words and thus a greater number of bytes, functions of the number of bytes in a document could possibly be used for normalization. The Okapi system successfully uses the document size (in bytes) for length normalization of (correct) documents. [7] In OCR environments, the byte sizes of the documents are less distorted, and this distortion is much more uniform across documents. For this reason, byte sizes of documents should provide a more stable normalization function. [14]

We use byte size of a document to denote the document's length in the pivoted normalization function. Using the average byte size as the pivot, we obtain the following normalization function:

$$pivoted\ byte\ size\ normalization = (1 - slope) \times average\ byte\ size + slope \times byte\ size$$

Using this normalization function, which we denote by the letter b in Smart's notation, and $1 + log(tf)$ weighted term frequency factors, we tested various slope values on the correct TREC disks one and two, using TREC queries 151–200. The results of using *lnb* weighted documents and *ltb* weighted queries are shown in Table 4.

Table 4 shows that pivoted byte size normalization also yields important improvements over cosine normalization. It is mildly worse than using the best pivoted unique normalization on the correct text.

Cosine	Pivoted Byte Size Normalization			
	Slope			
	0.25	**0.30**	0.35	0.40
6,526 0.2840	6,634 0.3258	**6,678** **0.3277**	6,689 0.3261	6,570 0.3088
Improvement	+14.7%	**+15.4%**	+14.8%	+8.7%
Improvement over best (0.3361) Pivoted Unique	- 3.1%	**- 2.5%**	- 3.0%	-8.1%

Table 4: Estimation of a good slope in pivoted byte size normalization for TREC queries 151–200. Each entry shows the total number of relevant documents retrieved (out of 9,805) for all fifty queries, and the non-interpolated average precision. The improvements in average precision over cosine normalization and over pivoted unique normalization are also shown. The pivot value was set to 2,730, which is the average number of indexable bytes in a document for TREC disks one and two.

When we compare the probability of retrieval using the pivoted byte size normalization to the probability of relevance for documents, we observe that pivoted byte size normalization retrieves very long documents with lower chances than their chances of relevance. This can be fixed by using a milder normalization function (like $bytesize^{0.80}$ or $bytesize^{0.60}$) with a stronger (higher) slope. Very small improvements (less than one percent) were obtained using these milder normalization functions. Overall, pivoted byte size normalization is an effective way to normalize, and it will be especially useful in degraded text collections.

7 Conclusions

This study shows that if documents of all lengths are retrieved with similar chances as their likelihood of relevance, retrieval effectiveness improves. Pivoted normalization is a powerful technique to make any normalization function weaker or stronger, thereby reducing the systematic deviation in the retrieval probabilities of documents (retrieved using the normalization scheme) from their likelihood of relevance. Substantial improvements are achieved by pivoting the cosine normalization function. This study also observes the weakness of the cosine function for very long documents and proposes a fix — pivoted unique normalization. The byte size of documents can also be pivoted to obtain another effective document length normalization function.

References

[1] James Allan. Relevance feedback with too much data. In Edward Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 337–343. Association for Computing Machinery, New York, July 1995.

[2] J. Broglio, J.P. Callan, W.B. Croft, and D.W. Nachbar. Document retrieval and routing using the INQUERY system. In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 29–38. NIST Special Publication 500-225, April 1995.

[3] Chris Buckley. The importance of proper weighting methods. In M. Bates, editor, *Human Language Technology*. Morgan Kaufman, 1993.

[4] Chris Buckley, James Allan, Gerard Salton, and Amit Singhal. Automatic query expansion using SMART : TREC 3. In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 69–80. NIST Special Publication 500-225, April 1995.

[5] J.P. Callan. Passage–level evidence in document retrieval. In W. Bruce Croft and C.J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310. Springer-Verlag, New York, July 1994.

[6] D. K. Harman. Overview of the third Text REtrieval Conference (TREC-3). In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 1–19. NIST Special Publication 500-225, April 1995.

[7] S.E. Robertson and S. Walker. Some simple effective approximations to the 2–poisson model for probabilistic weighted retrieval. In W. Bruce Croft and C.J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241. Springer-Verlag, New York, July 1994.

[8] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC–3. In D. K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. NIST Special Publication 500-225, April 1995.

[9] Gerard Salton. *Automatic text processing—the transformation, analysis and retrieval of information by computer*. Addison-Wesley Publishing Co., Reading, MA, 1989.

[10] Gerard Salton, James Allan, and Chris Buckley. Approaches to passage retrieval in full text information systems. In Robert Korfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58. Association for Computing Machinery, New York, June 1993.

[11] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[12] Gerard Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill Book Co., New York, 1983.

[13] Gerard Salton, A. Wong, and C.S. Yang. A vector space model for information retrieval. *Journal of the American Society for Information Science*, 18(11):613–620, November 1975.

[14] Amit Singhal, Gerard Salton, and Chris Buckley. Length normalization in degraded text collections. Technical Report TR95-1507, Department of Computer Science, Cornell University, Ithaca, NY 14853, April 1995.

[15] Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley. Document length normalization. Technical Report TR95-1529, Department of Computer Science, Cornell University, Ithaca, NY 14853, July 1995.

[16] Howard Turtle. *Inference Networks for Document Retrieval*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, MA 01003, 1990. Available as COINS Technical Report 90-92.