On Retrieving Legal Files: Shortening Documents and Weeding Out Garbage

Scott Kulp and April Kontostathis
Department of Mathematics and Computer Science
Ursinus College
Collegeville PA 19426
sckulp@gmail.com;akontostathis@ursinus.edu

Abstract

This paper describes our participation in the TREC Legal experiments in 2007. We have applied novel normalization techniques that are designed to slightly favor longer documents instead of assuming that all documents should have equal weight. We have also developed a new method for reformulating query text when background information is provided with an information request. We have also experimented with using enhanced OCR error detection to reduce the size of the term list and remove noise in the data. In this article, we discuss the impact of these effects on the TREC 2007 data sets. We show that the use of simple normalization methods significantly outperforms cosine normalization in the legal domain.

1 Introduction

One of the problems people in the legal profession face is the discovery of relevant documentation. When preparing cases for trial, law firms must search through hundreds of thousands of documents in order to find the most pertinent information in support of their case. In the legal domain, recall is considered to be more important than precision, but an increase in precision at top ranks would prevent time wasted on irrelevant materials.

In response to this need, the Text Retrieval Conference (TREC) started a new "Legal" track in 2006. The IIT Complex Document Information Processing test collection was used as the data set for the 2006 and 2007 competitions. This collection consists of roughly 7 million documents (approximately 57 GB of uncompressed text) taken from the Legacy Tobacco Document Library hosted by the University of California at San Francisco. These documents were made public during various legal cases involving US tobacco companies as part of the settlement agreement.

The documents in the TREC Legal corpus have widely disparate lengths, from a few sentences to hundreds of pages. Our primary experiments looked at the impact that document normalization has on this type of corpus. The normalization techniques we used are described in Section 2.3. We believe that the normalization studies we have done can be generalized to other collections with widely varying lengths. In addition to a standard query, the TREC Legal topics include background information which can be helpful for improving search. We have devised a method for using this background information to better identify which query terms serve as the best discriminators. These techniques are described in Section 2.4. We also experimented with using OCR error detection algorithms to reduce the size of the term list and reduce noise in the data, as described in Section 2.2. The results of our studies appear in Section 3.

2 Approach

In this section we describe our search and retrieval platform and discuss the methodologies used in our experiments.

2.1 The Information Retrieval System

The search and retrieval system we used for this project is custom-built. The TREC Legal data set is large, and generating and storing a single term-by-document matrix for the entire index was infeasible with the resources available. We developed a way to split up the data set so that our system could index and run queries on each piece separately, but produce results as if the system processed the entire set at once. The approach mirrors generalized vector space retrieval. We had intended to use the log-entropy weighting scheme [6] for term weighting. However, we discovered a programming error after our runs were submitted and we ended up using only local weighting (log tf) for all runs. We compared log tf to log entropy in subsequent experiments and the precision and recall numbers are almost identical, which is very interesting. More analysis is needed to determine the effect of term weighting with power normalization, but this is left as future work. We used the following algorithm to index TREC Legal:

- Split up the original data set into some number of smaller pieces; indexing TREC Legal on a machine with 8 GB of RAM required us to split it up into 81 subsets, each approximately 700-750 MB in size.
- Index each subset. For each subset, we removed terms that appear less than six times from the index. The documents were also processed using Optical Character Recognition (OCR) software. We applied the OCR error detection algorithm described in [9], supplemented by some additional rules, to reduce the size of the term list. Table 1 shows the number of terms pruned with and without OCR error detection.
- Loop through each sub-index file, keeping a running total of all the global term frequencies. After all the global frequencies of the terms in each of the sub-indices are found, the global term weights could be calculated (but this step was inadvertently omitted in our experiments as explained above). Given enough memory resources on the computer, this step can be combined with the previous step.
- With the global term weight list loaded, each sub-index is again reloaded separately. The local weight (and global weight, if applicable) is then applied to each entry in the term-by-document matrix. The sub-indices are then re-saved.

After the data set is indexed, a list of queries is processed by running the queries separately against each sub-index and keeping a running list of the top-scoring documents for each query. In our experiments, the normalization of documents was done at query run time, because we changed it with each run, but normally normalization would be applied during the term weighting step above.

Our test computer has an Intel Core 2 Quad processor at 2.4 GHz with 8 GB of system RAM and is running Windows XP x64 Professional. Given some parallelization, it takes about 6-8 hours to index TREC Legal, 2 hours to process queries on the entire data set, and 45 minutes to run queries on just the documents that have been judged for relevance.

2.2 OCR Error Detection

The documents in the TREC Legal data set were scanned in using Optical Character Recognition (OCR) software. However, OCR technology is imperfect and often creates errors within the documents. For example, the word "wonderful" may be mistakingly read as "wonolerful" by the OCR software. Sometimes, such as when it comes across a graphic image, the OCR software generates garbage strings. Both mistakes can adversely affect the weighting of terms in the documents, as well as make the size of the index much larger than it should be.

To help alleviate this problem, our system can automatically detect OCR errors and remove them. We began by mimicking the garbage detection rules found in the rmgarbage system [9], and then added additional rules in order to find more items. The changes we have made to rmgarbage were done in order to remove terms more liberally, thus shrinking the index even more. We used the following rules to decide if a string is garbage (an example follows each rule):

- If a string is more than 20 characters in length, it is garbage. This rule was taken from rmgarbage, but shortened from 40 to 20 characters.

  Example: iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii...
- If the number of punctuation characters in a string is greater than the number of alphanumeric characters, it is garbage. This rule was taken from rmgarbage.

  Example: ?3//la'
- Ignoring the first and last characters in a string, if there are two or more different punctuation characters in the string, it is garbage. This rule was taken from rmgarbage.

  Example: b?bl@bjk.1e.322
- If there are three or more identical characters in a row in a string, it is garbage. This rule was taken from rmgarbage, but shortened from four or more characters to three.

  Example: aaaaaBlE

Table 1: Number of Terms Removed Using Pruning Methods

	Num Terms Removed
Prune <= 5 (No OCR Detect)	454,892,023
Prune <= 5, rmgarbage	3,160,618,688
Prune <= 5, rmgarbage ext	3,195,632,736

- If the number of uppercase characters in a string is greater than the number of lowercase characters, and if the number of uppercase characters is less than the total number of characters in the string, it is garbage. This is a new rule we developed when we saw that OCR errors often created excessive numbers of uppercase characters, but normally, in English, there is usually no more than one uppercase character in a term. However, sometimes real English words appeared in all uppercase characters, which is acceptable, so words that contain only uppercase characters are not considered garbage.

  Example: BBEYaYYq

- If all the characters in a string are alphabetic, and if the number of consonants in the string is greater than 8 times the number of vowels in the string, or vice-versa, it is garbage. This rule was taken from rmgarbage, but the threshold was shortened from 10 to 8.

  Example: jabwqbpP

- If there are four or more consecutive vowels in the string or five or more consecutive consonants in the string, it is garbage. This is a new rule we developed when we noticed that real English words with these traits are rare, but this property appeared often in OCR errors.

  Example: buauub

- If the first and last characters in a string are both lowercase and any other character is uppercase, it is garbage. This rule was taken from rmgarbage.

  Example: awwgrapHic

Table 1 lists the number of terms pruned using three different methods. These numbers count repeated terms as well (for example, if the misspelled term "lavvyer" appears five times, it would be counted five times). The total number of terms in the entire collection is 7,358,393,244 before any pruning has been done. We estimate that an full index for TREC Legal without pruning would uses about 62 GB of hard drive space. Using OCR error detection saves about 8 GB of space.

2.3 Document Normalization

One of the major problems in information retrieval is the way large documents have a natural advantage over shorter documents when processing queries, simply because there are more opportunities for term matches in longer documents. The goal of document normalization is to reduce this advantage so that small relevant documents have the same probability of being returned as large relevant documents. The most widely-used normalization method is cosine normalization, and we used it as the basis for comparison in our experiments. Some alternate normalization techniques include pivoted document length normalization [7], maximum tf normalization [6], BM25 [5], and byte length normalization [8]. Pivoted document length normalization and BM25 have been shown to be more effective compared to cosine normalization on some collections; however, these techniques require extensive training, so they are not directly comparable to our approach (although our approach could be enhanced with pivoting). Maximum tf weighting schemes [6] use the largest term weight in a document as the normalization factor. This approach does not seem applicable to the TREC legal collection, where we have such widely differing document lengths and the risk of having outliers (terms with weights that are much higher than the norm), due to undetected OCR errors or uncommon proper names, appears to be particularly high. Byte length normalization has been shown to be effective on collections which have OCR errors corrected and on collections with simulated OCR errors [8]. This technique does not directly apply to our situation because a significant number of the OCR errors in TREC Legal were removed by our preprocessor and the remaining OCR errors are quite real, not simulated. We plan to compare our results to the byte length normalization, BM25, and pivoted document length normalization in the future.

2.3.1 Cosine Normalization

The standard method of document normalization is cosine normalization. For every document, the normalization factor is calculated. The normalization factor is defined by the expression

$$\sqrt{\sum_{i=1}^{n} w_i^2}$$

where w_i is weight of the i^{th} term in the document, and n is the number of unique terms in the document. The original term-document weight is divided by this normalization factor to get the final normalized term-document weight. When this is applied to every document in the collection, each document will have length of one.

The problem with cosine normalization is that it assumes that the probability of relevance is completely independent from document length. However, it is more likely that very long documents do have a slightly higher chance of being truly relevant to a query, since they have more content. To account for this, we developed document normalization schemes that bestow less of a penalty on the longest documents.

2.3.2 Log Normalization

We first use a log function to normalize documents. Let t be the total number of terms in a document. The log normalization factor is defined by the expression

$$log(t)$$

The original term-document weight is divided by the normalization factor to find the final normalized term-document weight. We chose the log function because of its slow growth as t becomes higher. This way, while all documents are shortened somewhat, very long documents are not penalized as much as shorter documents.

2.3.3 Power Normalization

We also experimented with using different powers of t as the normalization factor.

Let t be the total number of terms in a document. The square root normalization factor is defined by the expression

$$\sqrt{t}$$

The cube root normalization factor is defined by the expression

$$t^{1/3}$$

The fourth root normalization factor is defined by the expression

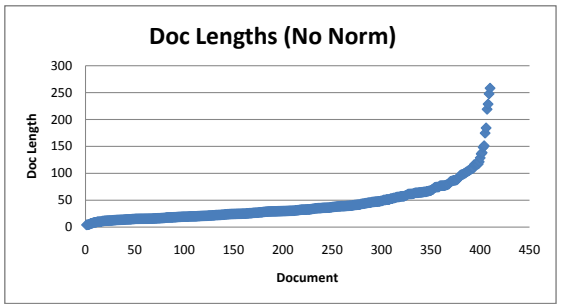
$$t^{1/4}$$

We used these powers of t for reasons similar to log normalization. These functions grow slowly with very large values of t, and so very large documents still have some advantage. The biggest difference between log normalization and power normalization is simply the rate of growth of the normalization factor functions. The power functions grow much faster than the log function, meaning the length advantage of extremely large documents is diminished more.

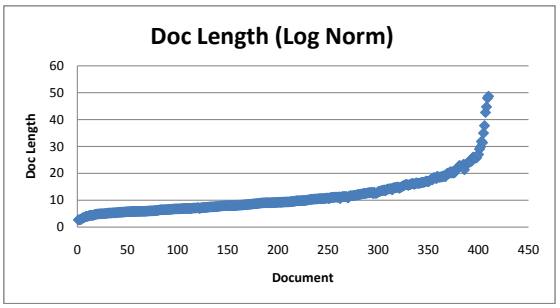
2.3.4 Analysis

Early experiments showed that log normalization worked well on small datasets with homogeneous lengths, but gave too much advantage to longer documents when used for the TREC Legal experiments. Figure 1 contains the data plots displaying the vector lengths of all the judged documents for the 2006 TREC Legal queries after normalization. A graph of document lengths after cosine normalization is applied would be a horizontal straight line at 1. The documents are sorted by vector length before normalization, in ascending order, so that document x in Figure 1(a) is referring to the same document as document x in Figure 1(b), as well as Figure 1(c), etc. In Figure 1(a), we can see the original document lengths, without normalization (sorted shortest document to longest document). On Figure 1(b), we can see the document lengths after log normalization is applied. The shape of the curve for log normalization is very similar to the original document length curve, meaning the log function had little effect.

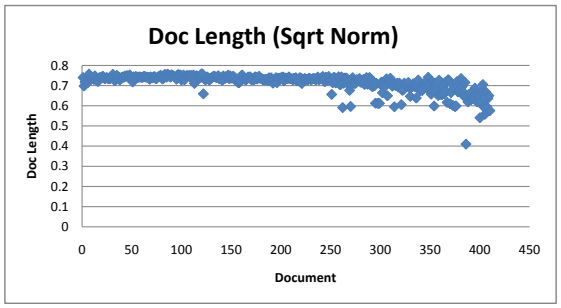
This prompted us to try power normalization techniques for TREC Legal. As we can see from Figures 1(c), 1(d) and 1(e), the power functions have a much greater effect on the curve. Figure 1(c) shows that the square root



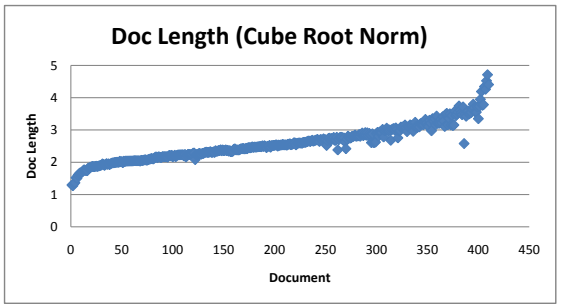
(a) No Normalization.



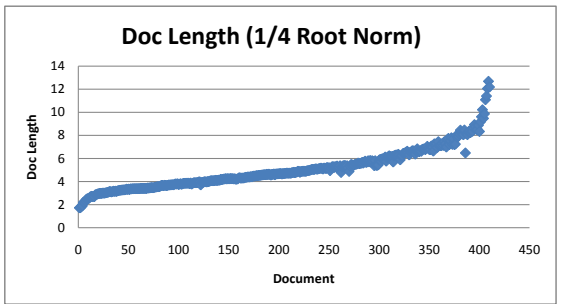
(b) Log Normalization.



(c) Square Root Normalization.



(d) Cube Root Normalization.



(e) Fourth Root Normalization.

Figure 1: The Effect of Normalization on Document Length

Table 2: Average Number of Terms in Returned Documents - 2006 queries

	No Norm	Log Norm	FrthRt Norm	CubeRt Norm	Sqrt Norm	Cosine Norm
Rank 10	194658.17	114085.49	11222.43	1709.70	362.09	299.68
Rank 20	174553.65	107315.34	12910.54	2011.27	434.03	348.11
Rank 30	162117.71	104196.73	14813.93	2118.35	458.15	398.22

Table 3: Average Number of Terms in Returned Documents - 2007 queries

	No Norm	Log Norm	FrthRt Norm	CubeRt Norm	Sqrt Norm	Cosine Norm
Rank 10	36691.7	28742.4	15845.5	8186.0	2146.5	2829.7
Rank 20	29168.3	24734.4	16084.3	11068.2	3093.4	3797.5
Rank 30	23651.3	21442.8	16207.4	11049.5	3381.9	4434.2

function is so heavy, the graph is actually somewhat downward-sloping. However, when using cube root or fourth root normalization, the graphs acts precisely as we intended with a very slight upward slope as original document sizes become longer.

In Tables 2 and 3, we can see the effect the normalization scheme has on the number of terms in the documents returned by our search system at top ranks for the 2006 and 2007 queries. As we expected, using no normalization results in too many large documents, which is not useful. Since log normalization does not have a very large effect on the long documents in TREC Legal, the average term length of the returned documents is within the same order of magnitude as using no normalization. Cosine normalization and square root normalization return very short documents. However, cube root normalization and fourth root normalization have average term counts that are between cosine/square root normalization and log normalization.

It is interesting to compare the 2006 numbers to the 2007 numbers. Although the trends are the same, the variation is much more pronounced for the 2006 queries. This leads us to believe that the 2007 queries are somehow different from the 2006 queries (collectively), but more analysis is needed to identify the differences. Subsequent experiments with the 2006 and 2007 queries separately have determined that retrieval performance also differs significantly [4].

2.4 Query Pruning

Another problem we faced in the TREC Legal project was deciding what to use as the queries. In the past, research has been done on expanding queries using such methods as thesaurus expansion [3] and automatic/manual query relevance feedback expansion [2, 1]. However, instead of expanding queries to make them longer, we wanted a way to make queries shorter and more specific.

In for each of the 46 topics in the 2006 set released by TREC legal, there is a short problem summary (the *request text*), and then a much longer overview of the problem. The problem summary is similar to a standard query, in that it is a simple one-sentence description of what the user wants to find. The following is an example:

All documents referencing the lobbying efforts of antismoking groups which specifically refer to their use of cartoons.

Using a standard English stop list, the terms *all, the, of, which, to, their,* and *use* are removed. However, the terms *documents, referencing, specifically,* and *refer* are jargon and are of no use when trying to do a document search. In fact, using these terms in the queries could have a negative effect on recall and precision, since documents that have matches to just those terms are probably not relevant.

To solve this problem, we needed a custom stop list to prune the legal jargon from the queries. However, given the very short length of the problem summaries, there was not enough information to automatically generate this stop list. We then decided to look at the longer problem descriptions. We used these descriptions directly as queries, but they were much too large (many were multiple pages long) and had too much extra information. They sharply lowered recall and precision rates when tested with the 2006 queries. They also are filled with technical jargon, and so we used them to automatically generate a stop list of high-frequency legal terms.

We developed a program which read all the topics. The system kept a record of the running total for each term's frequency throughout the query topics. We reviewed the list of terms that appeared more than sixty times in the query file to identify candidates for the legal term stop list, all but three of these terms were chosen for the stop list (the three that were eliminated were *California, media* and *code* - these were eliminated from the stop list because they did not appear to be legal jargon). Four additional terms were added to the legal stop list, as a result of manual analysis of the problem summary statements. The these four terms were *expressly, discussing, referencing, relating*. A legal term stop list consisting of 155 terms was created. In the next section we discuss our 2007 runs and see the effect of using this stop list on precision and recall rates.

3 Results

We were restricted to eight runs for 2007. The runs are described in Table 4. We selected runs that would test the effectiveness of each of the three strategies. For example, we could test the effectiveness of the various normalization strategies by comparing the results of runs 1-5. The effectiveness of automated query pruning can be gauged by comparing the manual runs 7 and 8 to query pruning runs 2 and 5. The OCR could be evaluated by comparing runs 6 and 2.

Table 4: Description of submitted runs for TREC Legal 2007

Run Name	Automatic?	Normalization	Query Pruning?	OCR?
ursinus1	Yes	Fourth Root	Yes	No
ursinus2	Yes	Cube Root	Yes	No
ursinus3	Yes	Square Root	Yes	No
ursinus4	Yes	Log	Yes	No
ursinus5	Yes	Cosine	Yes	No
ursinus6	Yes	Cube Root	Yes	Yes
ursinus7	No	Cube Root	No	No
ursinus8	No	Cosine	No	No

Table 5: Actual Precision Comparisons for TREC Legal 2007

	Rank									
Run Name	**5**	**10**	**15**	**20**	**30**	**100**	**200**	**500**	**1000**	**MAP**
ursinus5	0.065	0.056	0.056	0.049	0.042	0.027	0.023	0.015	0.012	0.010
ursinus1	**0.335**	**0.286**	**0.259**	**0.234**	**0.209**	**0.139**	**0.095**	**0.058**	**0.038**	**0.084**
ursinus2	0.302	0.237	0.214	0.202	0.170	0.106	0.077	0.045	0.034	0.065
ursinus3	0.070	0.065	0.062	0.058	0.050	0.030	0.023	0.015	0.012	0.010
ursinus4	0.251	0.177	0.144	0.136	0.119	0.075	0.056	0.035	0.024	0.034
ursinus6	0.237	0.219	0.194	0.176	0.151	0.096	0.068	0.047	0.032	0.057
ursinus7	0.265	0.202	0.177	0.157	0.140	0.100	0.069	0.044	0.029	0.052
ursinus8	0.074	0.084	0.074	0.066	0.058	0.038	0.032	0.020	0.014	0.012

The results provided by NIST are shown in Tables 5, 6, and 7. The precision rates for power normalization runs are much higher than precision rates for the cosine normalization (baseline) run. At rank 10, there is a 415% improvement in precision when using fourth root normalization over cosine normalization, and there is a 365% improvement in precision when using cube root normalization over cosine normalization. Large improvements in precision performance can be seen in all other rankings as well, when using cube or fourth root normalization over cosine normalization. Figure 2 shows the recall comparison between the cosine normalization baseline and the fourth root, cube root, and log normalization schemes. Interestingly, log normalization performed better on the 2007 queries than it did on comparable runs using the 2006 queries. We speculate that this may be related to the pooling techniques used in the evaluation process. Log normalization resulted in longer documents being retuned at top ranks, and therefore more gray documents appeared in top ranks. The estimated gray at B for log normalization was .1305 as compared with .0088 for cosine, .0157 for fourth root, and .0084 for cube root. The estimated gray at rank 5 for log was .3070, which was much larger than all other runs.

The results for Query Pruning were mixed. Generally the Query Pruning with cube root normalization (run 2) outperformed manual querying with cube root normalization (run 7), but the opposite was true when cosine normalization was used (see the data for runs 5 and 8 for cosine normalization with and without query pruning, resp.). This provides further evidence that there are subtle differences between the 2006 and 2007 queries, and they cannot be used interchangably to compare retrieval results.

As expected OCR detection had little impact on retrieval performance. A comparison of run 6 to run 2 shows that no OCR detection (run 2) has slightly higher retrieval performance in many cases. We speculate that the aggressive OCR error detection scheme may be removing true keywords unintentionally.

4 Conclusions

We have described experiments using several new algorithms that improve the performance of a search and retrieval system in the legal domain. Log and power normalization are promising new methods of document normalization that aim to retrieve longer, more relevant documents in the top ranks when the corpus has widely disparate lengths. Thus

Figure 2: TREC Legal Estimate Recall Comparison

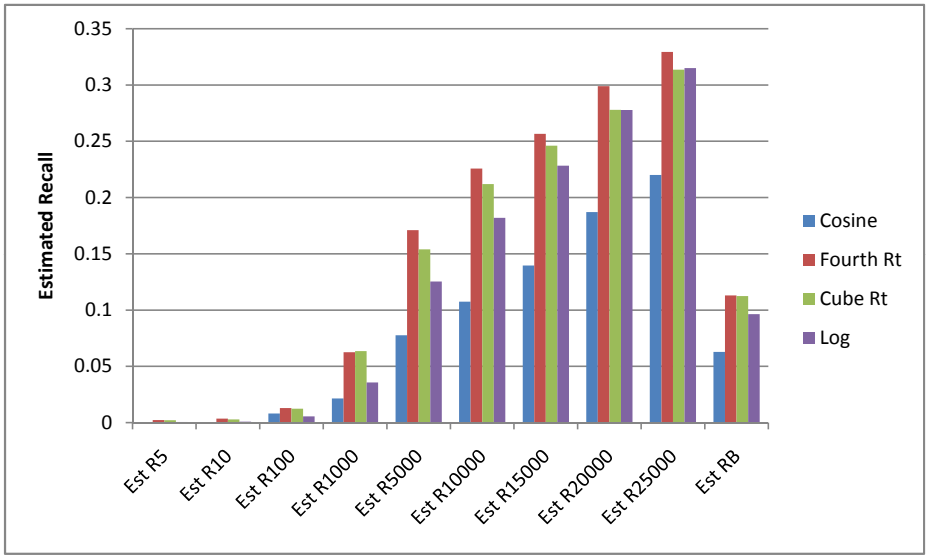


Table 6: Estimated Precision Comparisons for TREC Legal 2007

	Rank									
Run Name	**P5**	**P10**	**P100**	**P1000**	**P5000**	**P10000**	**P15000**	**P20000**	**P25000**	**Est P@B**
ursinus5	0.072	0.093	0.114	0.093	0.085	0.086	0.078	0.090	0.083	0.081
ursinus1	**0.340**	**0.365**	0.316	0.252	0.192	**0.188**	**0.165**	0.137	0.125	0.195
ursinus2	0.307	0.289	0.261	0.209	0.146	0.141	0.136	0.122	0.117	0.154
ursinus3	0.072	0.120	0.110	0.098	0.087	0.087	0.086	0.082	0.078	0.084
ursinus4	0.332	0.306	**0.321**	**0.293**	**0.204**	0.184	0.162	**0.151**	**0.139**	**0.233**
ursinus6	0.242	0.297	0.253	0.197	0.142	0.140	0.127	0.116	0.108	0.153
ursinus7	0.265	0.238	0.265	0.224	0.145	0.130	0.120	0.113	0.109	0.161
ursinus8	0.093	0.119	0.121	0.119	0.101	0.094	0.085	0.079	0.085	0.101

Table 7: Estimated Recall Comparisons for TREC Legal 2007

	Rank									
Run Name	**R5**	**R10**	**R100**	**R1000**	**R5000**	**R10000**	**R15000**	**R20000**	**R25000**	**Est R@B**
ursinus5	0.000	0.000	0.008	0.022	0.078	0.108	0.140	0.187	0.220	0.063
ursinus1	**0.002**	**0.004**	0.013	0.063	**0.171**	**0.226**	**0.257**	**0.299**	**0.329**	**0.113**
ursinus2	**0.002**	0.003	0.012	**0.064**	0.154	0.212	0.246	0.278	0.314	0.112
ursinus3	0.000	0.000	0.008	0.023	0.080	0.110	0.131	0.180	0.213	0.063
ursinus4	0.001	0.001	0.006	0.036	0.125	0.182	0.228	0.278	0.315	0.096
ursinus6	**0.002**	0.003	**0.016**	0.063	0.155	0.212	0.243	0.273	0.298	0.110
ursinus7	0.001	0.001	0.010	0.054	0.112	0.176	0.200	0.260	0.283	0.099
ursinus8	0.000	0.000	0.004	0.036	0.089	0.125	0.158	0.168	0.191	0.071

far, our testing has been restricted to the legal domain, but we believe similar improvements would be seen in other applications.

References

[1] Efthimis N. Efthimiadis. A User-Centred Evaluation of Ranking Algorithms for Interactive Query Expansion. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 146–159, New York, NY, USA, 1993. ACM Press.

[2] Susan Gauch and John B. Smith. Search Improvement via Automatic Query Reformulation. *ACM Transactions on Information Systems*, 9(3):249–280, 1991.

[3] Y. Jing and W. Bruce Croft. An Association Thesaurus for Information Retrieval. In *Proceedings of RIAO-94, 4th International Conference "Recherche d'Information Assistee par Ordinateur"*, New York, US, 1994.

[4] Scott Kulp. Improving Search and Retrieval Performance through Shortening Documents, Detecting Garbage, and Throwing Out Jargon. Technical report, Ursinus College, Collegeville, PA, USA, http://webpages.ursinus.edu/akontostathis/KulpHonorsThesis.pdf, 2007.

[5] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30, 1992.

[6] Gerard Salton and Chris Buckley. Term Weighting Approaches in Automatic Text Retrieval. Technical report, Cornell University, Ithaca, NY, USA, 1987.

[7] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted Document Length Normalization. In *Research and Development in Information Retrieval*, pages 21–29, 1996.

[8] Amit Singhal, Gerard Salton, and Chris Buckley. Length Normalization in Degraded Text Collections. Technical report, Cornell University, Ithaca, NY, USA, 1995.

[9] Kazem Taghva, Tom Nartker, Allen Condit, and Julie Borsack. Automatic Removal of "Garbage Strings" in OCR Text: An Implementation. In *The 5th World Multi-Conference on Systemics, Cybernetics and Informatics*, 2001.