How Effective Is Suffixing?

Donna Harman*
Lister Hill Center for Biomedical Communications, National Library of Medicine, Bethesda, MD 20209

The interaction of suffixing algorithms and ranking techniques in retrieval performance, particularly in an online environment, was investigated. Three general purpose suffixing algorithms were used for retrieval on the Cranfield 1400, Medlars, and CACM test collections, with no significant improvement in performance shown for any of the algorithms. A failure analysis suggested three modifications to ranking techniques: variable weighting of term variants, selective stemming depending on query length, and selective stemming depending on term importance. None of these modifications improved performance. Recommendations are made regarding the uses of suffixing in an online environment.

Introduction

Traditional statistically based keyword retrieval systems have been the subject of experiments for over 30 years. The use of simple keyword matching as a basis for retrieval can produce acceptable results, and the addition of ranking techniques based on the frequency of a given matching term within a document collection and/or within a given document adds considerable improvement (Sparck Jones, 1972; Salton, 1983).

The conflation of word variants using suffixing algorithms was one of the earliest enhancements to statistical keyword retrieval systems (Salton, 1971), and has become so standard a part of most systems that many system descriptions neglect to mention the use of suffixing, or to identify the algorithm was used. Suffixing was originally done for two principle reasons: the large reduction in storage required by a retrieval dictionary (Bell, 1979), and the increase in performance due to the use of word variants. Recent research has been more concerned with performance improvement than with storage reduction.

The NLM IRX (Information Retrieval Experiment) project (Benson, Goldstein, Fitzpatrick, Williamson, & Huntzinger, 1986; Harman, Benson, Fitzpatrick, Huntzinger, & Goldstein, 1988) has been investigating how the performance bounds of traditional statistically based keyword retrieval systems might be extended. Some of the initial research was in ranking algorithms, both in determining what factors are important in ranking, and in effectively combining these factors to maximize performance (Harman, 1986). Subsequently, work has been done using the most common method of query/document enhancements: the conflation of word variants using stemming. This area was selected both because of the widespread use of stemming, and because the interaction of ranking techniques and stemming techniques has not received much experimental attention. Word stems have been used interchangably with full words, and little effort has been made to modify the ranking techniques to handle the problems posed in ranking when using a stemmer. The flexibility of the IRX retrieval system permits easy modification of the ranking routines to use all information from a query, allowing an investigation into the limits of retrieval using suffixing.

*Current address: National Institutes of Standards and Technology, Gaithersburg, MD 20899.

Received June 1, 1988; revised January 4, 1989; accepted January 9, 1989. Not subject to copyright within the United States. Published by John Wiley & Sons, Inc.

Stemming Algorithms

Most research done on suffixing has been in the area of producing new suffixing algorithms, particularly algorithms aimed at specific collections or subject areas. An excellent review of suffixing approaches was done for the OKAPI project (Walker & Jones, 1987). Major work has been done in the area of medical English (Pacak, 1978). A technique for developing a stemming algorithm for a specific collection was devised for CITE, the NLM end-user interface to the CATLINE book catalog file (Ulmschneider & Doszkocs, 1983). Another technique for specific corpuses, which was developed by Hafer and Weiss (1974), used statistical properties of successor and predecessor letter variety counts. The goal of this research was not the development of a new stemming algorithm, but to enhance the performance of existing algorithms.

Three algorithms, an "*S*" stemming algorithm, the Lovins (1968) algorithm, and the Porter (1980) algorithm, were selected for this research effort because they have different levels of word conflation, a variable that was expected to be significant in stemming performance. Although many other stemming algorithms exist, these three all have widespread usage in the information retrieval community.

The "*S*" stemming algorithm, a basic algorithm conflating singular and plural word forms, is commonly used for minimal stemming. The rules for a version of this stemmer, shown in Table 1, are only applied to words of sufficient length (three or more characters), and are applied in an order dependent manner (i.e., the first applicable rule encountered is the only one used). Each rule has three parts; a specification of the qualifying word ending, such as "ies"; a list of exceptions; and the necessary action.

The SMART system (Salton, 1971) uses an enhanced version of the Lovins stemmer that removes many different suffixes. This stemming algorithm operates in much the same manner as the *S* stemmer, although at a higher level of complexity. First, the longest possible suffix is found that allows the remaining stem to be of length 2 or greater. The resulting word stem is then checked against an exception list for the given suffix, and, if passed, is processed into the final stem in a cleanup step. This cleanup step uses a set of rules to produce the proper word ending, such as removing a double consonant. The algorithm uses auxiliary files containing a list of over 260 possible suffixes, a large exception list, and the cleanup rules.

Porter (1980) developed a stemming algorithm that looks for about 60 suffixes, producing word variant conflation intermediate between a simple singular-plural technique and Lovins algorithm. The multistep process uses a successive removal of short suffixes, rather than a single removal of the longest possible suffix. Fewer suffixes are recognized, and no exception list is checked, resulting in both a much simpler algorithm than the Lovins algorithm, and no auxiliary files for the suffixes and their accompanying exceptions list.

As an example of stemmer differences, consider the word "heating," which has no "*S*" stems, and therefore is not grouped together with any other words using this stemmer. If the Porter stemmer is used, the word "heating" stems to "heate," which also happens to be the stem of the word "heated." Therefore these two words are grouped together for retrieval purposes. The Lovins algorithm stems the word "heating" to "heat," and groups "heat," "heated," "heater," "heating," and "heats" together, as all these words stem to "heat" using this stemmer.

In terms of storage, the "*S*" stemmer reduced the number of unique terms for the Cranfield collection from 8460 full words to 7489 unique stems. The Porter stemmer had a reduction from 8460 to 6028, and the Lovins stemmer reduced the number of unique concepts from 8460 to 5226. This could be important in an environment with limited storage, or for operational systems with large text files, because the length of the inverted files would be reduced as the number of unique terms is reduced.

TABLE 1. The *S* stemmer.

```
IF   a word ends in "ies," but not "eies" or "aies"
  THEN "ies" ⟶ "y"
IF   a word ends in "es," but not "aes," "ees," or "oes"
  THEN "es" ⟶ "e"
IF   a word ends in "s," but not "us" or "ss"
  THEN "s" ⟶ NULL
```

Modifications to IRX to Handle Suffixing

The IRX retrieval system used in this study parses each query into noncommon terms. When a stemming algorithm is used, the noncommon terms are mapped to a list of all words contained in the document collection that have the same word stem (as defined by the particular stemming algorithm in effect). These additional word variants are then added to the query, and a list of all documents containing one or more of the query terms serves as input to the ranking algorithm.

The ranking algorithm uses term weighting; that is, a document is ranked according to a normalized score formed by summing the weights of the terms occurring in the document that match terms in the query. This is equivalent to a weighted inner product between the query and the document. These weights are based on both the importance of the term within a given document (as measured by the $\log_2$ of the frequency of the term in that document), and on the importance of a term within the entire collection (as measured by either its noise value or its inverse document frequency weight). The ranking equations are given in the Appendix, and details of the ranking method can be found in Harman (1986).

Modifications to the ranking algorithm were required in order to reflect the traditional suffixing philosophy that all term variants for a given term be treated as a single concept. That is, the words "heat," "heats," "heater," "heating," and "heated" are traditionally represented by single concept (using the Lovins stemming algorithm) and term occurrences are counted for that concept rather than for the individual terms. The ranking algorithm was modified in two ways to handle suffixes in this manner. First, the document frequency of an expanded term in a given document becomes the combined document frequencies for all term variants of that term in that document. Second, instead of using the noise and number of postings for a given term, as determined by input from the parser, a special file of noise values and number of postings had to be created for each stemmer to obtain the correct noise and number of postings. This file was created by calculating the noise measure based on the term distribution of all term variants (instead of a single term), and by calculating the number of postings for all term variants, rather than for a single term. The ranker had to do a file lookup for each query term to determine its noise and number of postings. These modifications allowed IRX to emulate traditional suffixing. The alternative would have required separate inverted files for each stemmer.

Evaluation

Abstracts and titles from the Cranfield collection (with 225 queries and 1400 documents), comprised the major test collection for this study. The Medlars collection (30 queries and 1033 documents), and the CACM collection (64 queries and 3204 documents) were used to provide information about the variation of stemming performance across different subject areas and test collections.

In addition to the standard recall/precision measures, with SMART system averaging (Salton, 1971), several methods more suited to an interactive retrieval environment were adopted. The interactive environment returns lists of the top ranked documents, and allows the users to scan titles of a group of documents a screenful at a time, so that the ranking of individual documents within the screenful is not as important as the total number of relevant titles within a screen. Furthermore, the number of relevant documents in the first few screens is far more important for the user than the number of relevant in the last screenfuls. Three measures were selected which evaluate performance at given rank cutoff points, such as those corresponding to a screenful of document titles.

The first measure, the *E* measure (Van Rijsbergen, 1979), is a weighted combination of recall and precision that evaluates a set of retrieved documents at a given cutoff, ignoring the ranking within that set. The measure may have weights of 0.5, 1.0, and 2.0 which correspond, respectively, to attaching half the importance to recall as to precision, equal importance to both, and double importance to recall. A lower *E* value indicates a more effective performance.

A second measure, the total number of relevant documents retrieved by a given cutoff, was also calculated. Cutoffs of 10 and 30 documents were used, with ten reflecting a minimum number a user might be expected to scan, and 30 being an assumed upper limit of what a user would scan before query modification.

The third measure applicable to the interactive environment is the number of queries that retrieve no relevant documents by the given cutoff. This measure is important because many types of query modification techniques, such as relevance feedback, require relevant documents to be in the retrieved set to work well. These measures were all used in Croft (1983) as complementary measures to the standard recall/precision evaluation.

Results

Initial Stemming Results

The results for the Cranfield collection are given in Table 2. It can be seen that there is a very significant increase in performance (42.4%) from a simple ranking technique (number of term matches between the queries and the documents) to a technique using term weighting (full words and the noise ranking algorithm, see Appendix). None of the three suffixing techniques, however, achieve any further significant improvement over term weighting, using either the noise measure or the inverted document frequency measure for term weighting. This agrees with earlier results (Lennon, Peirce, Tarry, & Willett, 1981) using the Cranfield collection (titles only), and comparing the Lovins and Porter algorithms, when very little improvement in retrieval performance was found. Both the noise measure and the inverse document frequency weight measure were used in this study to show that the research results apply using either measure of term distribution within an entire collection.

The Medlars collection (Table 3) indicates a 22.2% improvement in performance from a simple ranking tech-

TABLE 2. Retrieval performance for Cranfield 225.

	Number of Matches	Best Noise Ranking				Best idf Ranking			
		Full Words	Lovins	Porter	*S*	Full Words	Lovins	Porter	*S*
Average precision for three intermediate points of recall	0.265	0.377	0.388	0.402	0.397	0.368	0.380	0.392	0.391
% Precision Change		42.4	45.2	49.2	47.7	39.1	41.3	45.7	45.4
E, 0.5, 10 docs	0.78	0.71	0.70	0.70	0.70	0.72	0.71	0.71	0.71
E, 1.0, 10 docs	0.77	0.68	0.68	0.68	0.68	0.70	0.69	0.69	0.68
E, 2.0, 10 docs	0.74	0.64	0.64	0.63	0.64	0.65	0.65	0.64	0.64
E, 0.5, 30 docs	0.87	0.84	0.83	0.83	0.84	0.84	0.84	0.84	0.84
E, 1.0, 30 docs	0.82	0.79	0.78	0.78	0.78	0.79	0.78	0.78	0.79
E, 2.0, 30 docs	0.73	0.67	0.66	0.67	0.67	0.68	0.67	0.67	0.67
Number of queries that fail									
At 10 documents retrieved	35	21	14	15	15	21	17	16	15
At 30 documents retrieved	16	9	8	8	8	10	7	6	8
Total relevant retrieved									
At 10 documents retrieved	473	650	655	666	654	628	636	648	651
At 30 documents retrieved	784	946	984	972	958	938	958	956	952

TABLE 3. Retrieval performance for Medlars collection.

	Number of Matches	Best Noise Ranking				Best idf Ranking			
		Full Words	Lovins	Porter	*S*	Full Words	Lovins	Porter	*S*
Average precision for three intermediate points of recall	0.426	0.522	0.574	0.539	0.538	0.519	0.543	0.521	0.525
% Precision Change		22.6	32.5	25.9	25.6	21.9	26.4	22.1	22.9
E, 0.5, 10 docs	0.56	0.50	0.50	0.51	0.50	0.49	0.49	0.50	0.49
E, 1.0, 10 docs	0.66	0.60	0.60	0.61	0.61	0.60	0.60	0.61	0.60
E, 2.0, 10 docs	0.71	0.67	0.66	0.67	0.67	0.66	0.66	0.67	0.66
E, 0.5, 30 docs	0.62	0.55	0.53	0.55	0.55	0.57	0.56	0.57	0.56
E, 1.0, 30 docs	0.60	0.52	0.49	0.51	0.52	0.53	0.53	0.54	0.53
E, 2.0, 30 docs	0.55	0.46	0.43	0.45	0.46	0.48	0.47	0.48	0.47
Number of queries that fail									
At 10 documents retrieved	2	0	0	0	0	0	0	0	0
At 30 documents retrieved	1	0	0	0	0	0	0	0	0
Total relevant retrieved									
At 10 documents retrieved	167	188	190	185	187	191	194	187	192
At 30 documents retrieved	329	387	412	394	389	377	382	372	383

nique to a technique using term weighting. Again, no suffixing technique achieves significant performance improvement. The CACM collection (Table 4) shows the most improvement in performance for suffixing, but the improvements shown are not statistically significant.

The lack of meaningful improvement for stemming is not because the retrieval is unaffected by stemming. Table 5 shows the number of queries that have an improvement or decrement in performance as measured by the number of relevant documents retrieved by either the top ten documents or the top thirty documents. The noise ranking method shows great improvement over simple matching, with an average of 5 queries showing an improvement for each query showing a decrement. However, for all the stemming techniques, the number of queries showing improvements in performance is nearly equalled (and surpassed in some cases), by the number of queries showing degradation in performance. The only exception to this is the CACM collection's use of the Lovins and Porter stemmers, which correlates with the unusually high performance of these stemmers in the CACM collection. For all collections, the Lovins stemmer generally produces both more improvement and more degradation in performance than either the Porter stemmer or the "*S*" stemmer, with the Porter stemmer performance closer to that of the Lovins stemmer than to the "*S*" stemmer.

In addition to the averaged evaluation measures reported above, several queries were analyzed individually in order to understand how ranks of the individual documents were affected by the stemming algorithms.

TABLE 4. Retrieval performance for CACM collection.

	Number of Matches	Best Noise Ranking				Best idf Ranking			
		Full Words	Lovins	Porter	*S*	Full Words	Lovins	Porter	*S*
Average precision for three intermediate points of recall	0.166	0.304	0.318	0.319	0.323	0.305	0.316	0.317	0.317
% Precision Change		82.8	87.2	87.5	89.0	83.5	87.0	87.3	87.3
E, 0.5, 10 docs	0.83	0.74	0.72	0.72	0.73	0.74	0.71	0.72	0.73
E, 1.0, 10 docs	0.85	0.76	0.74	0.74	0.75	0.76	0.74	0.75	0.75
E, 2.0, 10 docs	0.84	0.74	0.73	0.73	0.74	0.74	0.73	0.74	0.74
E, 0.5, 30 docs	0.86	0.81	0.78	0.79	0.80	0.81	0.78	0.78	0.80
E, 1.0, 30 docs	0.84	0.78	0.75	0.75	0.77	0.78	0.75	0.75	0.77
E, 2.0, 30 docs	0.79	0.71	0.67	0.68	0.70	0.71	0.67	0.68	0.70
Number of queries that fail									
At 10 documents retrieved	20	13	15	14	15	13	15	15	15
At 30 documents retrieved	18	13	12	12	12	13	12	12	12
Total relevant retrieved									
At 10 documents retrieved	99	153	171	169	159	152	173	168	159
At 30 documents retrieved	201	278	324	313	293	276	321	310	292

TABLE 5. Changes on a query-by-query basis.

	Queries with Improvement in Performance		Queries with Decrement in Performance	
	Cutoff = 10	Cutoff = 30	Cutoff = 10	Cutoff = 30
Cranfield (225 queries)				
noise ranking vs. match ranking	118	104	21	24
full word vs. Lovins	51	63	44	33
full word vs. Porter	49	49	37	31
full word vs. S stemming	34	39	32	33
Medlars (30 queries)				
noise ranking vs. match ranking	17	20	7	3
full word vs. Lovins	8	13	9	8
full word vs. Porter	8	12	11	9
full word vs. S stemming	6	10	7	8
CACM (64 queries)				
noise ranking vs. match ranking	35	38	6	4
full word vs. Lovins	23	25	12	8
full word vs. Porter	20	23	10	11
full word vs. S stemming	12	16	7	10

Table 6 shows the expansion of terms in query 109 from the Cranfield collection by the various stemmers. For example, the "*S*" stemmer adds the words "panel" and "aerodynamics" to the query. Of the two terms added to the query by the "*S*" stemmer, only one, "panel," helps retrieval. This effect can also be seen for the other stemmers, with the Porter stemmer adding four useful terms and three nonuseful terms, and the Lovins stemmer adding five useful terms and six nonuseful terms. These nonuseful terms cause nonrelevant documents to have higher ranks, and therefore often lower the ranks of the relevant documents.

Query 109 illustrates the major problem using stemmers—nonrelevant documents receive higher ranking scores, often surpassing those of the relevant. The main reason for this is the large number of terms being effectively added to the query (the word variants). Table 7 shows the average number of terms in a query for the various collections, and the average effective number of terms added by each stemmer. The "*S*" stemmer, which only conflates singular/plural variants, adds the least terms, expanding the query only by a factor of 1.5 on average. The Porter stemmer produces more word variants for a given term, expanding the query by a factor of 3 on average, and the Lovins stemmer produces the largest number of word variants mapping to a given root, expanding the query by a factor of over 4 on average. The increased number of terms leads to a much larger number of documents being retrieved, and this requires greater discrimination from the ranking algorithm.

Experiments to Improve Ranking Performance Using Stemming

Reweighting Term Expansions. The first effort to improve performance was a set of experiments in adjusting

TABLE 6. Stemmer performance for query 109 of the Cranfield collection (Query—"panels subjected to aerodynamic heating").

Full Word	*S*	Porter	Lovins
panels[a]	panel[a]	panel[a]	panel[a]
	panels[a]	panels[a]	panels[a]
subjected	subjected	subjected	subjected
		subject[a]	subject[a]
		subjective	subjective
		subjects	subjects
aerodynamic[a]	aerodynamic[a]	aerodynamic[a]	aerodynamic[a]
	aerodynamics	aerodynamics	aerodynamics
		aerodynamically[a]	aerodynamically[a]
			aerodynamicist
heating[a]	heating[a]	heating[a]	heating[a]
		heated[a]	heated[a]
			heat[a]
			heats
			heater

[a]Terms that were found in the relevant documents, and therefore become useful retrieval terms for this specific query.

TABLE 7. Average query expansion by stemming.

	No Stemming	*S* Stemmer	Porter Stemmer	Lovins Stemmer
Cranfield				
effective number of terms	10	16	28	39
total number of documents retrieved	736	837	886	943
Medlars				
effective number of terms	11	18	28	40
total number of documents retrieved	296	356	398	444
CACM				
effective number of terms	13	22	47	58
total number of documents retrieved	894	1253	1403	1459

term weights of stemmed results. It seemed intuitively helpful to use the stemmers as recall devices, increasing the pool of potentially relevant documents, but modifying the ranking mechanism to enhance precision. One method for increasing precision could be to assign less weight to those terms added by the stemmer, since they presumably are not as precise as the word form used in the query. It is possible in the IRX system to assign different term weights to each word variant for a given root, rather than treating all variants as if they were the same term. Instead of grouping all word variants for a given term as described earlier, each term variant was just added to the query, with its individual number of postings, noise, and document frequencies.

An experiment was performed in which the variants of a term added by the stemmer were down-weighted. Two different downweightings were used: half the initial term weight, and one-quarter the initial term weight, along with no reweighting. This allowed the suffixing technique to retrieve the extra documents that contained terms that were word variants, but gave less weight to these terms in ranking.

Table 8 shows the results from these experiments. The first three columns repeat the results shown in Table 2 for the Lovins stemmer. The last three columns, labeled "no grouping," show the results using no grouping of the term variants, and the three reweighting schemes. There was no improvement in performance, either for the Cranfield collection using the Lovins stemmer, or for the other collections and stemmers (not shown). An examination of individual queries indicated two factors in the lack of performance improvement.

First, the grouping of stemmed terms allowed their individual document frequencies to be summed for a given document, and that sum scaled down using a $\log_2$ function. This means that each additional term variant in a document generally adds less to the total document weight for ranking purposes than a completely different term. Not grouping the term variants causes each term variant to count as much as a completely different term, and this is clearly not reasonable.

The second factor is the calculation of the noise measure. For a standard grouping of term variants into a single concept, the noise of that concept is always higher than any of the noise values for the individual term variants. This means that terms having many variants tend to have high noise values, and this automatically downweights term variants. Not grouping the term variants allows their individual lower noise values to be used, giving terms with many variants more relative weight than terms with fewer variants.

When this effect is combined with the higher document frequency counts for no grouping, the result is that docu-

TABLE 8. Downweighting term variants in the Cranfield 1400.

		Best Noise Ranking				
	Match Ranking	Full Words	Lovins Grouping	Lovins No Grouping No rw	Lovins No Grouping rw 1/2	Lovins No Grouping rw 1/4
Average precision for three intermediate points of recall	0.265	0.377	0.388	0.309	0.387	0.377
% Precision Change		42.4	45.2	16.5	45.8	42.2
Total relevant retrieved						
At 10 documents retrieved	473	650	655	545	648	644
At 30 documents retrieved	784	946	984	848	948	940

ments containing more term variants (even of the same term) have higher weights than those having matches with terms that have few variants. Grouping forces occurrences of term variants of a given term to count less than occurrences of multiple unique words, and this is important for retrieval performance. Downweighting term variants corrected this problem to some extent, but not enough to improve performance over traditional grouping methods. Heavy downweighting produced results similar to no stemming, as the variants added little to a document weight.

Selective Stemming Based on Query Length. A second effort to improve the suffixing performance was based on controlling the number of terms used for retrieval by using stemming for only short queries. Short queries, i.e., queries with fewer than a given number of (noncommon) terms, were expanded using the stemming algorithms, but no stemming was used on longer queries. Short queries were defined as queries having fewer than 10 terms, which account for about half (54%) of the Cranfield query collection. To further test the hypothesis, a second division of the collection was made in which 69% of the query collection was considered short (queries having fewer than 12 terms).

The use of the suffixing algorithms on only the shorter queries did not improve overall performance. The result (Table 9) using the Lovins stemmer only for short queries was slightly better than no stemming, and slightly worse than the initial runs using the Lovins stemmer for all queries. The other stemmers and collections behaved in a similar manner. The number of terms in the original query is not predictive of whether the query performance can be improved by stemming.

Selective Stemming Based on Term Importance. The third effort to improve performance was based on the hypothesis that adding the term variants of words that are already widespread in the database degrades performance, and stemming should be applied only to "important" terms in the database. Term importance was estimated by the term's distribution in the database, as expressed by the noise measure. Three runs were made at different noise thresholds, with only terms having a noise below that threshold being stemmed using the Lovins stemmer. The first threshold of 5.000 stemmed 93% of the unique terms, involving 47% of the postings. The second and third thresholds of 6.000 and 7.000 stemmed 97% and 98% of the unique terms, involving 63% and 70% of the postings, respectively.

The experiments with selective stemming did not improve performance (Table 10). The results were all slightly worse than full stemming, with performance approaching that of full stemming at the highest threshold (as expected). A closer examination of the queries revealed that using the distribution of a term in the database as the criteria for stemming that term is not satisfactory, as many terms important to retrieval for a given query have a high frequency and even distribution, and these terms need to be expanded by stemming.

Conclusion

The use of the three general purpose stemming algorithms did not result in improvements in retrieval performance in the three test collections examined, as measured by classical evaluation techniques. Although individual

TABLE 9. Selective stemming using query length in the Cranfield 1400.

		Best Noise Ranking			
	Match Ranking	Full Words	Lovins All	Lovins n Terms < 10	Lovins n Terms < 12
Average precision for three					
intermediate points of recall	0.265	0.377	0.388	0.383	0.384
% Precision Change		42.1	46.1	44.3	44.9
Total relevant retrieved					
At 10 documents retrieved	473	650	655	659	658
At 30 documents retrieved	784	945	984	960	966

TABLE 10. Selective stemming using term importance in the Cranfield 1400.

		Best Noise Ranking				
	Match Ranking	Full Words	Lovins All	Lovins < 5.000	Lovins < 6.000	Lovins < 7.000
Average precision for three						
intermediate points of recall	0.265	0.377	0.388	0.376	0.375	0.382
% Precision Change		42.1	46.1	41.9	41.6	44.0
Total relevant retrieved						
At 10 documents retrieved	473	650	655	643	644	652
At 30 documents retrieved	784	945	984	957	954	961

queries were affected by stemming, the number of queries with improved performance tended to equal the number with poorer performance, thereby resulting in little overall change for the entire test collection. Attempts to improve performance by changing the weighting of stemmed terms which did not occur in the query, or by restricting stemming to short queries or to "important" terms, did not affect overall performance.

Recommendations for the Use of Suffixing in Online Environments

It is important to consider the implications of these results in an online environment. First, there are some storage savings using stemming. For small databases on machines with little storage, a sizable amount of inverted file storage can be saved using stemming. A source text of 1.6 megabytes, with 2653 documents, needs an index of 0.47 megabytes (0.14 dictionary + 0.33 postings) for unstemmed terms, but only 0.38 megabytes (0.08 dictionary + 0.30 postings) using the Lovins stemmer. However, for the larger databases normally used in online retrieval, less storage is saved. Using a source text of 50 megabytes, the index of unstemmed terms requires 6.7 megabytes (1.7 dictionary + 5.0 postings), compared to 5.8 megabytes (1.5 dictionary + 4.3 postings) using Lovins stemmer, not enough difference to justify stemming for storage savings.

Second, and more important, system performance must reflect a user's expectations, and the use of a stemmer (particularly the *S* stemmer) is intuitive to many users. The OKAPI project (Walker & Jones, 1987) did extensive research on improving retrieval in online catalogs, and strongly recommended using a "weak" stemmer at all times, as the "weak" stemmer (removal of plurals, "ed" and "ing") seldom hurt performance, but provided significant improvement. They found drops in precision for some queries using a "strong" stemmer (a variation of the Porter algorithm), and therefore recommended the use of a "strong" stemmer only when no matches were found. This implies some type of selective stemming should be used, but the batch-oriented research reported in this paper showed the difficulty of doing this automatically. One method of selective stemming is the availability of truncation in many online commercial retrieval systems. However, Frakes (1984) found that automatic stemming performed as well as experienced user truncation, and most user studies show little actual use of truncation.

Given today's retrieval speed and the ease of browsing a ranked output, a realistic approach for online retrieval would be the automatic use of a stemmer, using an algorithm like Porter (1980) or Lovins (1968), but providing the ability to keep a term from being stemmed (the inverse of truncation). If a user found that a term in the stemmed query produced too many nonrelevant documents, the query could be resubmitted with that term marked for no stemming. In this manner, users would have full advantage of stemming, but would be able to improve the results of those queries hurt by stemming.

Appendix

Ranking Using the Noise Measure

$$\text{rank}_j = \sum_{k=1}^{Q} \frac{(\log_2 \text{Freq}_{jk} \times (\text{noise}_{\max} - \text{noise}_k))}{\log_2 M}$$

where
Q = the number of terms in the query
Freq_{jk} = the frequency of query term k in record j
$\text{noise}_{\max}$ = the maximum value of noise_k for a given database
M = the total number of significant terms (including duplicates) in the record j (length factor)

$$\text{noise}_k = \sum_{i=1}^{N} \frac{\text{Freq}_{ik}}{\text{TFreq}_k} \log_2 \frac{\text{TFreq}_k}{\text{Freq}_{ik}}$$

where
N = the number of records in the database
Freq_{ik} = the frequency of term k in record i
TFreq_k = the total frequency of term k in the database

Ranking Using the IDF Weighting Measure

$$\text{rank}_j = \sum_{k=1}^{Q} \frac{(\log_2 \text{Freq}_{jk} \times \text{IDF}_k)}{\log_2 M}$$

where
Q = the number of terms in the query
Freq_{jk} = the frequency of query term k in record j
IDF_k = the inverse document frequency weight of term k for a given database
M = the total number of significant terms (including duplicates) in the record j (length factor)

$$\text{IDF}_k = \log_2 \frac{N}{\text{Num } D_k} + 1$$

where
N = the number of records in the database
$\text{Num } D_k$ = number of documents in the collection that contain one or more instance of term k

Acknowledgments

Thanks to the rest of the IRX project team, Dennis Benson, Larry Fitzpatrick, Charles Goldstein, and Rand Huntzinger, who made this research possible. In particular, thanks to Larry Fitzpatrick for his patient help with UNIX and to Dennis Benson and Charles Goldstein for their many helpful comments. Peter Willett provided the excellent reference to the OKAPI project.

References

Bell C., & Jones, K. P. (1979). Toward everyday language information retrieval systems via minicomputers. *Journal of the American Society for Information Science, 30,* 334–338.

Benson D., Goldstein C. M., Fitzpatrick L., Williamson D., & Huntzinger R. (1986). Developing tools for online medical reference works. In *Proceedings of Medinfo 86.* (pp. 558–560). Amsterdam: Elsevier Science Publishers B. V.

Croft W. B. (1983). Experiments with representation in a document retrieval system. *Information Technology: Research and Development, 2,* 1–21.

Frakes, W. B. (1984). Term conflation for information retrieval, In C. J. van Rijsbergen (Ed.), *Research and development in information retrieval* (pp. 383–389). Cambridge University Press.

Hafer M., & Weiss S. (1974). Word segmentation by letter successor varieties, *Information Storage and Retrieval, 10,* 371–385.

Harman D. (1986). An experimental study of factors important in document ranking, In *Proceedings of the 1986 ACM Conference on Research and Developments in Information Retrieval* (pp. 186–193). Pisa, 1986.

Harman D., Benson D., Fitzpatrick L., Huntzinger R. & Goldstein C. (1988). IRX: An information retrieval system for experimentation and user applications. In *Proceedings of the 1988 RIAO Conference.* Cambridge, Mass.

Lennon M., Peirce D., Tarry B., & Willett P. (1981). An evaluation of some conflation algorithms for information retrieval. *Journal of Information Science, 3,* 177–188.

Lovins J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics, 11,* 22–31.

Pacak M. G., & Pratt A. W. (1978). Identification and transformation of terminal morphemes in medical english, Part II, *Methods of Information in Medicine., 17,* 95–100.

Porter M. F. (1980). An algorithm for suffix stripping. *Program, 14,* 130–137.

Salton G. (1971). The *SMART retrieval system–experiments in automatic document processing.* Englewood Cliffs, N.J: Prentice-Hall.

Salton G., & McGill M. (1983). *Introduction to modern information retrieval.* New York: McGraw-Hill.

Sparck-Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation, 28,* 11–20.

Ulmschneider J. & Doszkocs T. (1983). A practical stemming algorithm for online search assistance. *Online Review, 7,* 301–315.

VanRijsbergen C. J. (1979). *Information retrieval.* London: Butterworths.

Walker, S., & Jones, R. M. (1987). *Improving subject retrieval in online catalogues.* British Library Research Paper 24.