A Detailed Analysis of English Stemming Algorithms

Article · March 1996

Source: CiteSeer

CITATIONS
51

READS
793

2 authors, including:

Gregory Grefenstette
Florida Institute for Human and Machine Cognition
157 PUBLICATIONS **4,932** CITATIONS

SEE PROFILE

A Detailed Analysis of English Stemming Algorithms

David A. Hull **Gregory Grefenstette**

Rank Xerox Research Centre
6 chemin de Maupertuis, 38240 Meylan France
{hull,grefen}@xerox.fr

January 31, 1996

Abstract

We present a study comparing the performance of traditional stemming algorithms based on suffix removal to linguistic methods performing morphological analysis. The results indicate that most conflation algorithms perform about 5% better than no stemming, and there is little difference between methods in terms of average performance. However, a detailed analysis of individual queries indicates that performance on this level is often highly sensitive to the choice of stemming technique. From this analysis, we can suggest a number of different ways to modify linguistic approaches so that they will be better suited to the stemming problem.

1 Introduction

In information retrieval (IR), the relationship between a query and a document is determined primarily by the number and frequency of terms which they have in common. Unfortunately, words have many morphological variants which will not be recognized by term-matching algorithms without additional text processing. In most cases, these variants have similar semantic interpretations and can be treated as equivalent for information retrieval (as opposed to linguistic) applications. Therefore, stemming or conflation algorithms have been created for IR systems which reduce these variants to a root form.

The linguistics groups at Xerox[1] have developed a number of linguistic tools for English which can be used in information retrieval. In particular, they have produced an English lexical database which provides a morphological analysis of any word in the lexicon and identifies the base form. There is good reason to expect that this technology would be ideally suited for use as a stemming algorithm. However, this assumption needs to be tested by conducting experiments using IR test collections.

In this paper, we present a detailed analysis of the impact of stemming algorithms on performance in information retrieval. We compare traditional approaches based on suffix removal to linguistic methods based on the Xerox morphological tools. We provide a detailed analysis which identifies specific examples of when the different methods succeed or fail. On average, there is not a lot of difference between stemming algorithms, but for specific queries, the choice of conflation stratégy can have a large impact on performance.

[1] Natural Language Theory and Technology (NLTT) at the Xerox Palo Alto Research Center and Multi-Lingual Theory and Technology (MLTT) at the Rank Xerox Research Center in Grenoble, France

2 Background - The Stemming Problem

The problem of conflation has been approached with a wide variety of different methods, as detailed in Lennon [10], including suffix removal, strict truncation of character strings, word segmentation, letter bigrams, and linguistic morphology. Two of the most popular algorithms in information retrieval, the Lovins stemmer [11] and the Porter stemmer [13], are based on suffix removal. Lovins finds the longest match from a large list of endings while Porter uses an iterative algorithm with a smaller number of suffixes and a few context-sensitive recoding rules.

Krovetz [9] accurately describes the problems associated with these methods. Most stemmers operate without a lexicon and thus ignore word meaning, which leads to a number of stemming errors. Words with different meanings are conflated to the same stem and words with similar meaning are not conflated at all. For example, the Porter stemmer conflates *general, generous, generation*, and *generic* to the same root, while related pairs like *recognize* and *recognition* are not conflated.

In addition, stems produced by suffix removal algorithms are often not words, which makes it difficult to use them for any purpose other than information retrieval. Interactive techniques which require user input, such as term selection for query expansion, will suffer greatly if the user must work with stems instead of real words. It also becomes difficult to perform dictionary look-up without real words. While the entries in the dictionary could themselves be stemmed, there would not be a one-to-one correspondance between word stems and word definitions. Dictionary look-up is an important feature in many IR applications. For example, in multilingual information retrieval, the query may be subject to automatic translation, in which case the system must be able to find each query term in a transfer dictionary. The problems cited in this section can easily be overcome by an approach to stemming which uses morphological analysis.

3 Xerox Lexical Technology

Morphology is a branch of linguistics that studies and describes how words are formed in language and includes inflection, derivation, and compounding. Inflection characterizes the changes in word form that accompany case, gender, number, tense, person, mood, or voice. Derivational analysis reduces surface forms to the base form from which they were derived, and includes changes in the part of speech.

Xerox linguists have developed a lexical database for English (as well as other languages) which can analyze and generate inflectional and derivational morphology. The inflectional database reduces each surface word to the form which can be found in the dictionary, as follows [15]:

- nouns $\Longrightarrow$ singular (ex. children $\Longrightarrow$ child)
- verbs $\Longrightarrow$ infinitive (ex. understood $\Longrightarrow$ understand)
- adjectives $\Longrightarrow$ positive form (ex. best $\Longrightarrow$ good)
- pronoun $\Longrightarrow$ nominative (ex. whom $\Longrightarrow$ who)

The derivational database reduces surface forms to stems which are related to the original in both form and semantics. For example, government stems to govern while department is not reduced to depart since the two forms have different meanings. All stems are valid English terms, and irregular forms are handled correctly. The derivational process uses both suffix and prefix removal, unlike most conventional stemming algorithms which rely solely on suffix removal. A sample of the suffixes and prefixes that are removed is given below [15]:

- suffixes: ly, ness, ion, ize, ant, ent, ic, al, ic, ical, able, ance, ary, ate, ce, y, dom, ee, eer, ence, ency, ery, ess, ful, hood, ible, icity, ify, ing, ish, ism, ist, istic, ity, ive, less, let, like, ment, ory, ous, ty, ship, some, ure

- prefixes: anti, bi, co, contra, counter, de, di, dis, en, extra, in, inter, intra, micro, mid, mini, multi, non, over, para, poly, post, pre, pro, re, semi, sub, super, supra, sur, trans, tri, ultra, un

The databases are constructed using finite state transducers, which promotes very efficient storage and access. This technology also allows the conflation process to act in reverse, generating all conceivable surface forms from a single base form. The database starts with a lexicon of about 77 thousand base forms from which it can generate roughly half a million surface forms [15]. The Derivational Analyzer is currently being used in the Visual Recall information access and retrieval product available from XSoft.

4 Description of previous work

There have been a large number of studies which have examined the impact of stemming algorithms on information retrieval performance. Frakes [4] provides a nice summary, reporting that the combined results of previous studies make it unclear whether *any* stemming is helpful. In those cases where stemming is beneficial, it tends to have only a small impact on performance, and the choice of stemmer among the most common variants is not important. However, there is no evidence that a reasonable stemmer will hurt retrieval performance.

In contrast, a recent study by Krovetz [9] reports an increase of 15-35% in retrieval performance when stemming is used on some collections (CACM and NPL). Krovetz mentions that these collections have both queries and documents which are extremely short, so that the results make sense given that the likelihood of an exact match of surface form should decrease with the size of the document. For collections with longer documents (TIME and WEST), stemming algorithms are accompanied by a much more modest improvement in retrieval performance.

Krovetz [9] also develops stemmers based on inflectional and derivational morphology. He finds that these approaches provide about the same improvements as were obtained by the Porter algorithm. However, he notes that the derivational stemmer performed slightly better than the inflectional stemmer over all four collection that he examined and that the majority of its benefit came at low recall recall levels (i.e. when relatively few documents are examined). We should mention that Krovetz's inflectional and derivational stemmers are not equivalent to the Xerox linguistic tools. In particular, he experiments with a smaller collection of suffixes for derivational analysis and it does not appear that he examines prefixes at all (perhaps wisely, as we shall discover). Therefore, his results are not directly comparable to the ones presented in this paper.

5 The Experimental Collection

Our retrieval experiments will use the document collection constructed for the TREC/TIPSTER project. The TREC collection consists of over a million documents (3.3 gigabytes of text) obtained from a variety of sources, including newspapers, computer journals, and government reports. As part of the exhaustive evaluation studies conducted during the TIPSTER project, analysts have constructed 200 queries and evaluated thousands of documents for relevance with respect to these queries. The collection is so large that we have decided to select a subset of the queries and/or documents to use in our experiments. Since experiments using stemmers require that the entire document collection be reindexed for each algorithm, storing many different indexed version of the full collection is not really feasible. Therefore, we have decided to use the

Wall Street Journal sub-collection, which consists of 550MB of text and about 180,000 articles for our experiments.

The TREC queries, generally called topics, are large and very detailed, and provide a very explicit definition of what it means for a document to be relevant. In fact, with an average of 130 terms per topic, they are comparable in length to the documents in the Wall Street Journal sub-collection (median length = 200 terms). The TREC experiments have frequently been criticized for the length of the topics, since this is in such marked contrast to user behavior when querying the typical commercial retrieval system, where queries of one or two terms are often the norm. While the precision and detail of TREC topics is extremely valuable for making accurate relevance judgements and encouraging researchers to develop complex retrieval strategies, the experimental results may not reflect the tools and techniques that will be most valuable for operational systems.

In order to address this issue, we have constructed shorter versions of the topics which attempt to summarize the key components of the query in a few short phrases (average length = 7 words). This follows the lead of Voorhees [14] and Jing [8] who use the description statement as a summary for the full query. In contrast to their approach, we construct the new queries by hand, as it was felt that some of the description statements were lacking important key words. There is certainly an element of subjectivity in this approach, but the queries were constructed without regard to the specific properties of stemming algorithms.

6 The retrieval system and stemming experiments

We used the SMART[2] text retrieval system developed at Cornell University [2] for the information retrieval experiments. We found its high indexing speed (500MB/hr on a Sparc 20) to be extremely valuable for the repeated indexing of the collection that is necessary in stemming experiments. Very frequent terms are removed using a slightly modified version of the SMART stop list, and queries and documents are treated as vectors of term frequencies and analyzed using the vector space model.

Term weighting is used to improve performance. Term frequencies in both queries and documents are dampened by applying the square-root transformation. Document vectors are normalized to have unit length and query term weights are multiplied by the traditional IDF (inverse document frequency = $\log N/n_i$, N = # docs in collection, n_i = # docs containing term i) measure to increase the importance of rare terms. No proper name or phrase recognition is used although these might well be beneficial. Also, no attempt is made to segment long documents, which fortunately are rare in the Wall Street Journal sub-collection.

We will examine the performance of five different stemming algorithms and compare them to a baseline which consists of no stemming at all. Two stemmers are included in the SMART collection; they are a simple algorithm which removes *s*'s from the end of the word and an extensively modified version of the Lovins algorithm [11]. In addition, we will study the Porter stemmer [13], and versions of the Xerox English inflectional and derivational analyzers [15] slightly modified for the conflation problem.

7 Experimental Results

The SMART system is used to index the queries and documents separately for each stemming algorithm. Documents are ranked according to their similarity to each query, and the results are evaluating using the IR measures precision (the fraction of retrieved documents that are

[2]Available for research purposes via anonymous ftp to ftp.cs.cornell.edu in directory /pub/smart.

Query	Measure	nostem	remove s	Lovins	Porter	Inflect	Deriv	sig. test
full	APR11	0.348	0.361	0.369	0.367	0.368	0.366	RLPID $\succ$ N
full	AP[5-15]	0.556	0.562	0.557	0.562	0.562	0.555	-
full	AR[50-150]	0.414	0.423	0.427	0.429	0.428	0.426	LPID $\succ$ N
short	APR11	0.179	0.198	0.209	0.206	0.201	0.208	RLPID $\succ$ N
short	AP[5-15]	0.313	0.339	0.343	0.356	0.345	0.354	RLPID $\succ$ N
short	AR[50-150]	0.263	0.282	0.288	0.290	0.285	0.288	RLPID $\succ$ N

Table 1: Average evaluation scores by stemming algorithm.

relevant) and recall (the fraction of all relevant documents that are retrieved[3]).

We choose to use three different evaluation measures, average precision at 11 recall points, 0, 10%, . . . 100% (APR11), average precision at 5-15 documents examined (AP[5-15]), and average recall at 50, 60, . . . 150 documents examined (AR[50-150]). The first measurement is the current evaluation standard in IR experiments and captures a wide range of different performance characteristics. The second measurement is designed to estimate performance for shallow searches and the third is chosen to capture a more in-depth inquiry by the user. The evaluation scores for the six stemming algorithms over 200 TREC queries are presented in Table 1.

In general, it appears that all stemmers are slightly better than no stemming at all but there is little difference between stemmers. This comes as no real surprise and agrees with the conclusions obtained from many previous studies. It is also interesting to note how much sharply cutting query size reduces information retrieval performance. However, the size of the query does not make a large difference in the relative performance of stemming algorithms.

The observed differences are relatively small, so it is important to validate them using statistical significance testing. Hypothesis tests make the preliminary assumption that all retrieval strategies are equally effective. The test determines the probability that the observed results could occur by chance (or p-value) given the initial hypothesis. If the p-value is very small, then evidence suggests that the observed effect reflects an underlying difference in performance. Statistical testing is important because the queries used for testing represent only a very small sample from the set of all possible queries. If the number of queries is small relative to the observed difference in evaluation scores, then the experimental results are not generally applicable.

Our approach to statistical testing is based on the Analysis of Variance (ANOVA), which is useful for detecting differences between three or more experimental methods. The two-way ANOVA assumes that the evaluation scores can modelled by an additive combination of a query effect and an effect due to the choice of experimental method. When these effects are factored out, it is assumed that the remaining component of the score is random error, and that all such values are drawn from an identical Normal distribution, independent of query or method. It is unlikely that these assumptions will be strictly accurate, but it is hoped that they represent a reasonable approximation. In order to run the ANOVA, the evaluation scores must be computed separately for each query and method, generating a table which is used as input to the statistical algorithm. For mathematical details on statistical testing in information retrieval, see Hull [6] or consult any elementary statistics textbook on experimental design, such as Neter [12].

The ANOVA test operates in two stages. First, a test statistic is computed which indicates whether there is any evidence of a difference between the methods. If the results indicate that there is a difference (in our experiments, we define a significant difference as one which produces a p-value less than .05), then a separate multiple comparisons test is applied to determine which

[3]The term *all relevant documents* is a little misleading, since it refers only to the relevant documents in the subset of the collection that has actually been examined. For TREC queries, this is usually thousands of the most likely documents, so it is unlikely that many relevant documents have been missed

Query	nostem	remove s	Lovins	Porter	Inflect	Deriv
Q178	0.228	0.170	0.486	0.486	0.191	0.169
ranks	4	2	5.5	5.5	3	1
Q187	0.127	0.129	0.117	0.124	0.132	0.188
ranks	3	4	1	2	5	6

Table 2: Example of within-query ranking

contrasts are significant. In this paper, we examine all pairwise differences in average scores. The results are presented in compact form in Table 1, and can be interpreted as shown below.

The expression x $\succ$ y indicates that method x is significantly better than method y according to the appropriate statistical test (ANOVA for the average scores, Friedman for the average ranks). The abbreviations are: N = nostem, R = remove s, L = Lovins, P = Porter, I = inflectional, D = derivational. For example, for APR11 using short queries, the entry is:

$$\text{RLPID} \succ \text{N, D}$$

which indicates that remove s, Lovins, Porter, Inflectional, and Derivational stemmers are all significantly better than no stemming. The results from the ANOVA make interpreting Table 1 very simple. As stated previously, all stemmers are better than no stemming, except for AP[5-15] measured on the full query, and there is no significant difference among stemming algorithms. A statistically significant difference is roughly 0.01 for most methods (0.023 for AP[5-15] short).

When the query is well defined and the user is only looking at a few documents (AP[5-15] full), stemming provides absolutely no advantage. This helps to explain a bit of the inconsistency in the literature. Lennon [10] evaluates at 5 and 20 documents retrieved and Harman [5] evaluates at 10 and 30 documents retrieved and find no difference between stemmers. Harman's average precision-recall scores look very similar to our own. Krovetz [9] reports much larger improvements for stemming in collections where both queries and documents are short, but the other collections show a similar 4-5% improvement in performance.

It seems natural to consider stemmers as recall enhancing devices, as they are creating more matches for each query term. Our results reflect this hypothesis, as stemming is no help at low recall and provides some benefits when performance is averaged over a number of recall levels. Stemming is slightly more helpful for short queriesand low recall, as demonstrated by the fact that no stemming leads to a drop in AP[5-15] for short queries but not for long ones. As Krovetz demonstrates, stemming becomes much more valuable when both queries and documents are short, as this is when it is most difficult to find term matches between query and document.

8 Evaluation based on Average Ranks

Unfortunately, it sometimes happens that average measurements do not adequately describe overall performance. For example, perhaps there exist one or a few queries which have a huge variability between methods, while for the others, the difference is much smaller. The average performance figures will be dominated by these queries and completely insensitive to other patterns in the data. For this reason, we present an alternative evaluation measure that is based on the ranking of şcores between methods for each query.

Table 2 presents the APR11 scores for two of the short queries. Clearly, query Q178 has more variability than query Q187. We can normalize for unequal variance by replacing the score for each method by its ranking with respect to the scores of other methods for the same query. Most people would naturally wonder if this is desirable. Surely Q178 is demonstrating a much more important difference than Q187 which is completely lost when the scores are ranked. The answer to this is maybe, but maybe not. In this example, Q178 has only 3 relevant documents while

Q187 has 429! Most of the variability in Q178 is accounted for by the change in ranking of a single relevant document. In this example, it could well be the case that the differences described by Q187 are more reliable. Since in our experiment, the number of relevant documents per query ranges from 2 to 591, there is good reason to believe that it might be worthwhile to examine an evaluation measure that normalizes for unequal variance[4]. Using the ranking strategy described above, the evaluation results can be summarized by taking the average rank over all queries, as presented in Table 3.

Query	Measure	none	rem s	Lov	Port	Infl	Der	sig. test
full	APR11	2.70	3.12	3.83	3.76	3.79	3.81	LPID $\succ$ R $\succ$ N
full	AP[5-15]	3.34	3.48	3.66	3.57	3.58	3.37	-
full	AR[50-150]	2.83	3.29	3.68	3.63	3.79	3.79	LID $\succ$ R $\succ$ N, P $\succ$ N
short	APR11	2.62	3.25	3.70	3.79	3.68	3.96	LPID $\succ$ R $\succ$ N
short	AP[5-15]	3.05	3.45	3.55	3.69	3.52	3.76	RLPID $\succ$ N
short	AR[50-150]	2.63	3.20	3.60	3.91	3.70	3.97	LPID $\succ$ R $\succ$ N, D $\succ$ L

Table 3: Average rank measurements

How does one compare average ranks and average precision-recall scores? It is hard to say, since the measures are on different scales. The disadvantage of ranks is that any sense of absolute performance is lost. The ranks also depend on which methods are being compared. Remove one stemming algorithm from the analysis and the ranks will change, perhaps significantly. However, there are a number of advantages as well. For example, it is now simple to compare between the different evaluation scores and also between short and long queries, since the average ranks of all evaluation measures are on the same scale. Note that average performance is 3.5 for a ranking based on six methods.

While the overall pattern is about the same, there are several noticeable differences between the average ranks for the full and short queries.

(1) AP[5-15] nostem - No stemming is proportionally less effective for short queries. We can thus conclude that stemming always improves performance for short queries.

(2) AP[5-15] Deriv - The derivational stemmer works better over the short queries, where it can help to find a match, than on the long queries, where it adds noise to the results.

(3) AR[50-150] Porter - The Porter stemmer appears to work better with the short queries than the long queries at high recall.

Since rank based analysis is only a relative measure, it should not be used as the only means of evaluation, but it provides a nice supplement to the data provided by average precision and recall.

The same approach to hypothesis testing used above can also be applied to the ranking data, and this alternative is known in the statistical literature as the Friedman Test. For more mathematical details, see Hull [6] or Conover [3]. The results are presented in Table 3. The Friedman test finds more significant differences than the ANOVA. In particular, it consistently finds that remove s is less effective than the other stemmers except for AP[5-15]. The magnitude of a significant difference in ranking is roughly 0.35. There are some puzzling inconsistencies in the statistical results. For example, the derivational stemmer is significantly better than the Lovins stemmer using AR[50-150] short query rankings, but the actual recall scores of the two methods are equal to three decimal places. We will come back to this result at the end of the next section.

[4]This problem is one of the unfortunate side-effects of using of subset of the TREC collection. If we used the full collection, we would have a lot less variation in the number of relevant documents for each query.

9 Statistical Testing

It is worth noting that the Friedman Test appears to be consistently more powerful than the ANOVA, in the sense that it finds more significant differences between methods. This is a bit surprising, since the Friedman Test is a nonparametric test (not model based), and tests from this family are usually less powerful than the ANOVA, when its assumptions are reasonable. This suggests that it would be a good idea to check those assumptions.

The ANOVA model assumes that the error variance is constant across queries. Given that each query has a widely variable number of relevant documents, there is good reason to expect that this condition may be violated. As we saw in the previous section, the evaluation score of queries with very few relevant documents may change dramatically with the change in rank of a single relevant document. We can examine the data more closely to see how badly the assumption of equal variance is violated.

Let us presume that the variance is constant and equal to the error variance obtained from the ANOVA model, which we will denote as σ^2. It is well known in statistics [1] that:

$$\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1}$$

where s^2 is the variance of a sample of size $n-1$ drawn from a normal population with variance σ^2. We can compute s^2 for each query and compare the distribution of the test statistic above to the appropriate chi-square reference distribution. In particular, we determine the expected value of maximum of a sample of 200 observations drawn from the chi-square distribution. If our data is actually drawn from this distribution, we would expect an average of one observation to equal or exceed this value. In practice, we find that between 11-18 observations exceed the expected maximum for each of our experiments.

The results above indicate that without question, the assumption of equal variance is violated. We have suggested that queries with few relevant documents will tend to have higher variability than those with many relevant documents. We examined this hypothesis and found that while a few of the queries with highest variability had fewer than 10 relevant documents, there was no consistent pattern. We also tested whether variability in performance was significantly correlated with average performance and also obtained a negative result. From this analysis, we believe that the results from the Friedman Test should be more reliable for our experiments. It also provides evidence that examining average rank data as well as average score data is valuable in determining which experimental methods are most effective.

The reader should keep in mind that the statistical tests described here have been chosen for this particular experiment. In general, the optimal test may depend on the number of experimental methods being compared. For instance, both Friedman and ANOVA have different forms when there are only two methods [6], and Conover [3] suggests that there is a more powerful alternative to the Friedman Test for cases when three or four methods are being compared.

10 Identifying Important Queries

In the previous section, we obtained strong evidence that variance is not constant over queries. Since these queries are much more important in determining the average performance of an experimental method than their low variance counterparts, it is valuable to get a better feeling for what causes this behavior. We have found three major causes for high variance in a query.

(1) A lot of relevant documents are ranked higher in some methods than in others.

(2) The query has very few relevant documents and thus scores are sensitive to changes in the ranking of one or two relevant documents.

Average Scores

Query	Measure	nostem	remove s	Lovins	Porter	Inflect	Deriv	sig. test
full	APR11	0.375	0.393	0.402	0.399	0.400	0.397	RLPID $\succ$ N
full	AP[5-15]	0.623	0.640	0.628	0.642	0.640	0.631	-
full	AR[50-150]	0.377	0.391	0.394	0.398	0.397	0.391	RLPID $\succ$ N
short	APR11	0.208	0.233	0.244	0.241	0.238	0.245	RLPID $\succ$ N
short	AP[5-15]	0.374	0.410	0.412	0.429	0.421	0.426	RLPID $\succ$ N
short	AR[50-150]	0.259	0.284	0.287	0.294	0.290	0.288	RLPID $\succ$ N

Average Ranks

Query	Measure	none	rem s	Lov	Port	Infl	Der	sig. test
full	APR11	2.50	3.19	3.82	3.74	3.88	3.88	LPID $\succ$ R $\succ$ N
full	AP[5-15]	3.24	3.52	3.50	3.64	3.58	3.51	-
full	AR[50-150]	2.66	3.32	3.78	3.70	3.84	3.70	RLPID $\succ$ N, LI $\succ$ R
short	APR11	2.51	3.24	3.58	3.90	3.85	3.92	PID $\succ$ R $\succ$ N, L $\succ$ N
short	AP[5-15]	2.94	3.44	3.52	3.70	3.69	3.71	RLPID $\succ$ N
short	AR[50-150]	2.54	3.22	3.40	3.99	3.89	3.97	PID $\succ$ LR $\succ$ N

Table 4: Evaluation scores for the important queries

(3) The query is hard and relevant documents tend to have low ranks in general. However, some methods rank a few relevant documents relatively well, which can make a large difference for many evaluation measures.

Clearly, factor (1) describes the type of behavior we are looking for and these queries deserve to be considered very important. Some people might argue that queries of type (2) and (3) are also important, because they demonstrate real changes in the ranks of relevant documents, however we believe that this is a dangerous conclusion. If only a few relevant documents are being ranked higher, it is hard to know if this is due to the fact that they are relevant or some reason unrelated to relevance. We would be particularly suspicious of type (3) queries, for one would hope that a method which is valuable would improve the ranking of a reasonably large sample of relevant documents. This does not mean that an effect which improves the rank of a single relevant document is not real and important in some cases, merely that this is a less reliable indicator of value than a method which improves the rank of many relevant documents.

Therefore, we would like to evaluate average performance using only queries which satisfy factor (1). This can be accomplished, by applying the Friedman Test on a per-query basis by comparing the ranks of individual relevant documents. A query of type (1) will have consistent differences in the ranks of the relevant documents, which can be detected by the Friedman Test. There are a number of issues to consider when applying this method which will not be discussed here. The reader interested in more details is encouraged to refer to Hull [7]. From this analysis, we obtain 125 full queries and 139 short queries which satisfy our definition of an important query. Average performance scores are computed over this subset and presented in Table 4. Readers are cautioned that these results are not unbiased estimates of the true differences between methods due to the selective nature of the query filtering process.

The significant differences increase to roughly 0.015 (0.032 for short AP[5-15]) for average scores and 0.42 for average ranks, due to the fact that fewer queries are being analyzed, however the results remain roughly the same. This reassures us that the potential problems that we discussed at the beginning of the chapter do not change the results a great deal. We will concentrate on the average rank statistics since they are directly comparable to the rank statistics computed using all queries.

For the full queries, the results are basically identical, although the average rank of no stemming drops even further relative to the other methods. This is probably a result of removing the queries where stemming is not important and indicates that when stemming is important, not stemming is rarely the right decision. For the short queries, there is another interesting pattern. The Lovins stemmer appears to be less effective while the Porter and Inflectional stemmer are proportionally more effective according to average rank. However, this pattern is not duplicated in the average scores, where the Lovins stemmer is fully competitive with the other stemmers. A closer look at the scores reveals that the Lovins stemmer performs slightly worse for a lot of queries and much better for a few queries, which explains the lower score with respect to the rank based measure.

One possible explanation is that Lovins stems more heavily than all the other stemmers. This means that it conflates a lot more terms, which reduces performance slightly on a lot of queries by adding extra noise. However, occasionally an additional important term is recognized, which helps performance a lot. Therefore, while average performance is the same, choosing the Lovins stemmer may degrade performance slightly for a lot of queries in exchange for helping more significantly with a few queries. Note that this hypothesis has not been verified explicitly and applies only to the short queries when evaluated using APR11 or AR[50-150].

11 Detailed Analysis

While we have learned a lot from our extensive analysis of the average performance figures, it has not really helped us find out exactly why one stemming algorithm works better than another. If we wish to improve the current technology in stemming algorithms, it is important to see specific examples where the type of stemming makes a large difference in performance. Therefore, we will take a detailed look at a number of individual queries and figure out which word stems account for differences in performance. For this section of the analysis, we rely only on the short queries, for it will be far easier to identify the important word stems in these examples.

At this point, it is very clear that no stemming and remove s are not really competitive with the other stemming algorithms, so they will not be considered further. Among the others we have found little evidence which suggests how to choose between them, other than a hint that the Lovins stemmer may have slightly different behavior. For the detailed query analysis, we start with only those queries which were judged important based on a per-query Friedman Test, as described in the previous section. We then compute the error variability of the queries and rank them in decreasing order of variability, just as we did when testing the equal-variance assumption for ANOVA, except now we are working only with four stemming algorithms. We compute the chi-square score for each query and find that 11 exceed the expected maximum score of 12.07 for a sample of 139 queries with equal variance. The top 10 of these are presented in Table 5.

The queries shown above can be divided into 6 categories according to their behavior.

(1) Lovins, Deriv $\gg$ Inflect, Porter - 21, 70, 82, 143

(2) Inflect, Porter $\gg$ Lovins, Deriv - 39, 147

(3) Deriv $\gg$ others - 86

(4) Deriv $\ll$ others - 182

(5) Porter $\ll$ others - 128

(6) Lovins, Porter $\gg$ Inflect, Deriv - 98

We examine each of these categories in more detail.

Q#	score	Lovins	Porter	Inflect	Deriv
86	61.37	0.216	0.225	0.230	0.546
39	57.27	0.092	0.343	0.365	0.094
21	53.5	0.596	0.302	0.278	0.484
82	39.23	0.438	0.282	0.151	0.406
143	26.11	0.199	0.101	0.007	0.249
128	24.87	0.246	0.047	0.251	0.255
182	24.42	0.305	0.341	0.262	0.115
98	24.37	0.501	0.507	0.326	0.331
147	20.27	0.032	0.186	0.187	0.034
70	15.9	0.635	0.496	0.495	0.647

Table 5: Revised evaluation scores

(1) Lovins, Deriv ≫ Inflect, Porter

Q21: superconductivity vs. superconductor

Unstemmed Word	Lovins	Porter	Inflect	Deriv
superconduct(ed/ing)	superconduc	superconduct	superconduct	conduct
superconduction	superconduc	superconduct	superconduction	conduct
superconductive	superconduc	superconduct	superconductive	conduct
superconductively	superconduc	superconductiveli	superconductively	conduct
superconductivity('s)	superconduc	superconductiviti	superconductivity	conduct
superconductor(s/'s)	superconduc	superconductor	superconductor	conduct

The query talks about research in *superconductivity* while many of the documents refer only to *superconductors*. Only the Derivational and the Lovins stemmers make this match. Note that the Derivational stemmer overstems by removing the prefix and so loses a bit in performance for this reason.

Q70: surrogate motherhood vs. surrogate mother

The query asks about surrogate *motherhood* while many of the documents talk about surrogate *mothers*. Only the Derivational and the Lovins stemmers make this connection.

Q82: genetic engineering vs. genetically engineered product

Unstemmed Word	Lovins	Porter	Inflect	Deriv
genetic('s)	genet	genet	genetic	genetic
genetically	genet	geneticalli	genetically	genetic
geneticist(s/'s)	genet	geneticist	geneticist	genetic
genetics('s)	genet	genet	genetics	genetics

The Derivational and Lovins stemmers relate *genetic* and *genetically*. The Inflectional stemmer lost out even more because *engineering* did not get conflated to *engineer*.

Q143: U.S. government protection of farming

Unstemmed Word	Lovins	Porter	Inflect	Deriv
farm(s/'s/ed)	farm	farm	farm	farm
farming('s)	farm	farm	farming	farm
farmer(s/'s)	farm	farmer	farmer	farm

The Inflectional and Porter stemmers did not recognize that *farmer* and *farming* were related. The Inflectional stemmer did not conflate *farming* and *farm* and thus was completely ineffective.

The behavior of the inflectional stemmer with respect to *farming* and *engineering* deserves a more complete explanation. Both words can be either a noun or a verb. According to inflectional rules, the noun form should not be stemmed while the verb form should. We do not apply a part-of-speech tagger to the text before stemming and are therefore forced to make an arbitrary decision. Our decision not to stem *engineering* and *farming* turns out to be a poor one for IR performance, although it may be the right one from the linguistic perspective in the absence of a part-of-speech tag.

(2) Inflect, Porter $\gg$ Lovins, Deriv

Q39: client-server architectures

The derivational and the Lovins stemmer equate *server* with *serve*. This is a very bad decision since *serve* is a common term used in a number of contexts and *server* has a much more specific meaning, particularly in the domain of computers and technology.

Q147: productivity statistics for the U.S. economy

The Derivational and the Lovins stemmer equate *productivity* and *produce*. One again, this turns out to be a bad decision for the reasons described in the previous example.

Note the contrasts between sections (1) and (2). Conflating *farmer* to *farm* is good but conflating *server* to *serve* is bad. *Superconductivity* should definitely be related to *superconduct* but converting *productivity* to *produce* turns out to be a bad decision. This clearly indicates that it is impossible to produce an ideal stemmer using only suffixing rules. There are only two ways to make a distinction in the examples above. One is to construct new rules or exception lists by hand, the other is to identify which conflation pairs are used in similar contexts by conducting a corpus-driven analysis.

(3) Deriv $\gg$ others

Q86: bank failures

The Derivational stemmer converts *failure* to *fail*. The other stemmers do not make this connection. This is an example where linguistic knowledge can recognize a special case which is missed by suffixing rules.

(4) Deriv $\ll$ others

Q182: commercial overfishing

The Derivational stemmer converts *overfishing* to *fish*. It is becoming increasing clear that prefix removal is a bad idea.

(5) Porter $\ll$ others

Q128: privatization of state assets

The Porter stemmer equates *privatization* with *private*. This is certainly unfortunate for information retrieval. The irony here is that the Derivational stemmer would probably have made the same mistake, but the word *privatization* is not in the lexicon! This points out another potentially serious problem that is shared by both the Inflectional and Derivational stemmer.

Since both stemmers analyze on the basis of linguistic rules, a word must be in the lexicon in order to be recognized.

We make the decision not to stem any word which does not appear in the lexicon. While this works well for the most part, since most unrecognized words are proper names, there is a distressingly large list of exceptions. This is a particular problem when using the Wall Street Journal, because a large amount of financial and economic terminology is not included in our general purpose lexicon. Many of these would be important when used as query terms. It might be a good idea to construct a guesser which applies a rules based algorithm when the word does not appear in the lexicon, because at the moment the linguistic stemmers do not even conflate plurals of unidentified nouns. We have implemented one special rule to remove possessives ('s) since this is obviously an important factor for proper names and the rule is extremely unlikely to make any errors.

(6) Lovins, Porter ≫ Inflect, Deriv

Q98: manufacturers of fiber optics equipment

The inflectional and derivational stemmers do not stem *optics* to *optic*. For the Derivational stemmer, this is a linguistically motivated decision which turns out to be very unfortunate for information retrieval.

The detailed query analysis has been very informative. We have recognized that there are many examples where the suffix-removal rules do not produce ideal behavior. We have also discovered that the Inflectional and Derivational stemmer make a number of decisions for linguistic reasons that are not correct for information retrieval. Also, the two linguistic stemmers could probably be improved by adding domain-specific lexicons or by building a guesser for unknown terms.

12 Modified Derivational Stemmer

The detailed query analysis has revealed that prefix removal is generally a bad idea for a stemming algorithm. The derivational stemmer suffers in performance on a number of occasions for this reason. For example, prefix removal for the terms *overfishing* (Q128) and *superconductor* (Q21) is certainly undesirable. A quick scan over the unexamined queries reveals several more instances where prefix removal causes problems: Q1 - *antitrust, illegal* and Q4 - debt *rescheduling.*

		Prefix Removal		
Query	Measure	Y	N	sig?
full	APR11	0.366	0.368	N
full	AP[5-15]	0.555	0.559	N
full	AR[50-150]	0.426	0.426	N
short	APR11	0.208	0.210	Y
short	AP[5-15]	0.354	0.356	N
short	AR[50-150]	0.288	0.289	N

Table 6: Effect of prefix removal on stemming performance

To verify this hypothesis, we printed out a list of all terms for which prefix removal took place in the Wall Street Journal. It quickly became obvious that in the vast majority of cases, stemming would be undesirable. A lot of prefixes, such as *anti-*, *un-*, and *il-* reverse the meaning of the root form. Therefore, we created a new version of the Derivational stemmer that only

modified the suffix of the word, in order to see whether this would make a significant difference in retrieval performance. Table 6 compares performance to the original version.

Query	Measure	none	rem s	Lov	Port	Infl	Der	sig. test
full	APR11	2.68	3.10	3.81	3.75	3.77	3.90	LPID $\succ$ R $\succ$ N
full	AP[5-15]	3.32	3.47	3.63	3.54	3.56	3.48	-
full	AR[50-150]	2.81	3.25	3.69	3.62	3.78	3.87	LPID $\succ$ R $\succ$ N
short	APR11	2.63	3.25	3.68	3.75	3.68	4.01	LPID $\succ$ R $\succ$ N
short	AP[5-15]	3.03	3.47	3.55	3.69	3.52	3.75	RLPID $\succ$ N
short	AR[50-150]	2.63	3.20	3.56	3.90	3.70	4.01	LPID $\succ$ R $\succ$ N, D $\succ$ L

Table 7: Ranks with revised Derivational stemmer

The average scores changes very little, but all the differences are in the right direction, so it seems to be worthwhile to use the modifications. To further examine this question, we replace the old Derivational stemmer with the new one and recompute the average ranks and the Friedman Test, as shown in Table 7. Both Table 6 and Table 7 are computed using the full query set. Although the revised derivational stemmer has slightly higher ranks, there is no significant change in the results. It might be worthwhile to look at only the important queries, be we did not run that experiment in time for this paper.

13 Conclusions

After an exhaustive analysis, we have reached the following conclusions concerning stemming algorithms.

(1) Some form of stemming is almost always beneficial. The only case where there is not overwhelming evidence in favor of stemming is when the full TREC queries are used and very few documents are examined. The average absolute improvement due to stemming is small, ranging from 1-3%, but it makes a large difference for many individual queries. There are probably particular queries where stemming hurts performance, but we did not investigate this issue in detail.

(2) Simple plural removal is less effective than more complex algorithms in most cases. However, when only a small number of documents are being examined, plural removal is very competitive with the other algorithms.

(3) There is no difference between the other stemmers in terms of average performance. For short queries, the Lovins stemmer tends to perform slightly worse for many queries but a lot better for a few others. This may be a result of overstemming although we have not produced any concrete evidence for this hypothesis.

(4) The detailed query analysis demonstrates that the same suffix-removal rule can be beneficial in some cases and harmful in others. This suggests that rules-based suffix removal may not be the ideal approach to stemming in the long run.

(5) There are a number of problems with using linguistic approaches to word analysis directly as stemming algorithms. First, these methods are based on a lexicon and cannot correctly stem words which are not contained in the lexicon. Second, many decisions about the root form of a word which are properly motivated from the linguistic perspective are not optimal for information retrieval performance. It is clear that linguistic analysis tools must be tailored for information retrieval applications.

(6) In particular, prefix removal seems to be a particularly bad idea for a stemming algorithm, and the detailed analysis provides a number of specific examples of this problem. Modifying the linguistic tools provides only a small benefit in terms of average performance, but the difference is consistent over all evaluation measures.

While the linguistically based stemming is not significantly better than current algorithms in term of average performance, the detailed query analysis reveals a number of different ways in which the linguistic tools could be improved and optimized for information retrieval. With such modifications, and their inherent advantages over suffix-removal algorithms (stems are real words), linguistic tools based on morphological analysis should work very successfully as stemming algorithms.

Acknowledgements We are grateful to Donna Harman and the analysts from the TIPSTER project for providing such an outstanding resource which has really helped to advance the field of IR. We would also like to thank Chris Buckley for writing and supporting the SMART text retrieval system and making it available for research use. This is a real service to the information retrieval community.

References

[1] G.E.P. Box, W.G. Hunter, and J.S. Hunter. *Statistics for Experimenters*, pages 118–119. John Wiley and Sons, 1978.

[2] Chris Buckley. Implementation of the smart information retrieval system. Technical Report 85-686, Cornell University, 1985.

[3] W.J. Conover. *Practical Nonparametric Statistics.* John Wiley and Sons, 2nd edition, 1980.

[4] W.B. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, 1992.

[5] Donna Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1):321–331, 1991.

[6] David Hull. Using statistical testing in the evaluation of retrieval performance. In *Proc. of the 16th ACM/SIGIR Conference*, pages 329–338, 1993.

[7] David A. Hull. Stemming algorithms - a case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1):70–84, 1996.

[8] Yufeng Jing and W. Bruce Croft. An association thesaurus for information retrieval. In *Proc. of Intelligent Multimedia Retrieval Systems and Management Conference (RIAO)*, pages 146–160, 1994.

[9] Robert Krovetz. Viewing morphology as an inference process. In *Proc. of the 16th ACM/SIGIR Conference*, pages 191–202, 1993.

[10] M. Lennon, D. Pierce, B. Tarry, and P. Willett. An evaluation of some conflation algorithms for information retrieval. *Journal of Information Science*, 3:177–183, 1981.

[11] Janet Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[12] J. Neter, W. Wasserman, and M. Kutner. *Applied Linear Statistical Models.* R.D. Irwin, 2nd edition, 1985.

[13] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[14] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *Proc. of the 17th ACM/SIGIR Conference*, pages 61–69, 1994.

[15] Xerox Corp. *Xerox Linguistic Database Reference*, english version 1.1.4 edition, December 1994.