Stemming and its effects on TFIDF Ranking

Mark Kantrowitz *mark@kantrowitz.com*
Behrang Mohit *behrang@andrew.cmu.edu*
Vibhu Mittal *mittal@cmu.edu*

Just Research
4616 Henry Street
Pittsburgh, PA 15213
U.S.A.

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
U.S.A.

1 Introduction

High precision IR is often hard for a variety of reasons; one of these is the large number of morphological variants for any given term. To address some of the issues arising from a mismatch between different word forms used in the queries and the relevant documents, researchers have long proposed the use of various stemming algorithms to reduce terms to a common base form. Stemming, in general, has not been an unmitigated success in improving IR. This poster argues that stemming can help in certain contexts and that an empirical investigation of the relationship between stemming performance and retrieval performance can be valuable. We extend previously reported work on stemming and IR (e.g., [2, 4, 5]) by using a novel, dictionary based "perfect" stemmer, which can be parametrized for different accuracy and coverage levels. This allows us to measure *changes in IR performance for any given change in stemming performance* on a given data set. To place this work in context, we discuss an empirical evaluation of stemming accuracy for three stemming algorithms – including the widely used Porter algorithm [9]. Section 2 briefly discusses the three variants of stemming, and presents experimental evidence for their relative coverage and accuracy. Section 3 discusses the use of these stemmers for IR and presents some of our findings. Finally, Section 4 concludes with a discussion of our results and possible future directions.

2 Stemmer Evaluation

To evaluate the quality of various stemmers, we generated 17,782 stem-equivalence classes containing 81,673 words. These equivalence classes are made up of words that stem to the same root token.[1] The word that most closely matched the root, such as the singular form of nouns and the infinitive form of verbs, was listed first. Usually this was the shortest word in the equivalence class. The stemmer was considered to have stemmed a word correctly if the result of the stemming was the first word in the equivalence class.

For stemming algorithms like Porter's stemmer, which generate a truncated pseudo-root instead of a word as the stem, the algorithm was considered to have stemmed a word correctly if the word's stem matched the result of applying the stemmer to the first word in the word's equivalence class. This establishes an upper bound on the stemmer's accuracy, since this evaluation method cannot detect an overly aggressive stemmer that merges equivalence classes by stemming their roots to the same pseudo-root. Porter's stemmer was found to merge 238 pairs of equivalence classes, or about 1.3% of the equivalence classes.

The set of equivalence classes can be used as a perfect stemmer by mapping each word to the first word in its equivalence class using a hash table or trie. Such a stemmer would have 100% accuracy, but only to the extent that the words to be stemmed were included in the equivalence classes. Furthermore, such a stemmer could be combined with a stemmer like Porter's or one of the other stemmers described below to improve their overall accuracy. Effectively the perfect stemmer caches the correct root for exceptions, turning to the other stemming algorithm for out-of-vocabulary words.

Since stemming and spelling correction can both be considered variants of lemmatization, we developed some alternate stemmers based on spelling correction algorithms, such as the Damerau-Levenshtein edit distance, but with the dictionary of valid words restricted to the words that are roots. Such algorithms offer an additional benefit, in that they can correctly stem even words with spelling and typographic errors. A cursory examination of a few dozen randomly selected web pages found that about 5% of the words were misspelled; in many of these cases, the typical spelling errors involved words like "suprised", "officals", "reprsents", and "inappopriately", which are not correctly stemmed by other stemming algorithms, despite the errors not appearing in the suffix. (Other benefits include stems that are actual words instead of pseudo-roots and ease of extension.)

Three stemming algorithms based on spelling correction algorithms were evaluated.

1. **Edit Distance:** uses edit distance to find the closest matching stem, with a preference for shorter stems by using edit costs that favor deletions over transpositions, insertions, and substitutions.

2. **Edit Distance with Complex Edits:** assigns zero cost to complex edits corresponding to the most common suffixes (e.g., deleting "ing" or substituting "e" for "able").

[1] The equivalence classes were compiled twice, with one pass going forward through the dataset, and a second pass going backward, with 99.74% agreement. The average number of words per equivalence class was 4.6.

3. **Prefix Stemmer:** records the longest common prefix (substring) for each stemming equivalence class and stems a word by finding the longest prefix that matched the word, returning the first word of that prefix's equivalence class as the stem.

We evaluated Porter's stemmer, two s-deletion stemmers (a simple s-deletion stemmer and Harmon's s-deletion stemmer [2]), and the spelling correction stemmers, obtaining the following results:

S-Deletion Stemmer	19.7%
Harmon's S-Deletion Stemmer	20.9%
Porter's Stemmer	66.1%
Edit Distance with Extra Knowledge	90.2%
Edit Distance w/o Extra Knowledge	93.4%
Prefix Stemmer	93.7%

It would appear that the three new stemming algorithms being used here are significantly more accurate than Porter's stemmer and the s-deletion stemmers.

We also evaluated Porter's stemmer and the three spelling correction stemmers on subsets corresponding to the most frequent English words.[2] The assumption was that irregular word forms are more common among the more frequent words, and so a hand-crafted stemmer like Porter's stemmer should do better. But as the following results demonstrate, the spelling correction stemmers always outperform Porter's stemmer even though Porter's performance improves on the more frequent words.

	Top 100	Top 1,000	Top 10,000
Porter's Stemmer	78.3%	80.2%	77.2%
Edit Distance with	80.8%	87.9%	90.4%
Edit Distance w/o	82.7%	89.6%	93.1%
Prefix Stemmer	79.9%	87.4%	92.2%

Table 1: Stemmer effectiveness

The similar performance of the edit distance and prefix stemmers suggests that much of the performance of these algorithms can be accounted for by the distinctiveness of the stem words from each other.

Thus we have found several stemmers that significantly outperform Porter's stemmer and s-deletion in terms of stemming accuracy. We then evaluated the improvement in precision of Porter's stemmer, the prefix stemmer, and the perfect stemmer with an unstemmed data set using the TREC-2 corpus with short and long queries.

3 Effect on Retrieval

In order to analyze the effect of stemming on retrieval performance, we ran a number of experiments on four different data sets.[3] The largest of these corpora was the TREC-2 corpus consisting of approximately 78,000 news wire articles. We also tested the validity of our results by testing on a variety of datasets, including the Cranfield-1400, the MED and the ADI data sets.

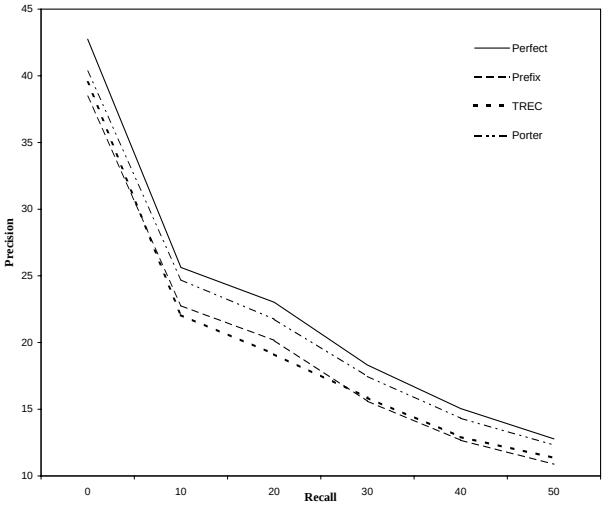

Figure 1: Short queries on the TREC-2 dataset.

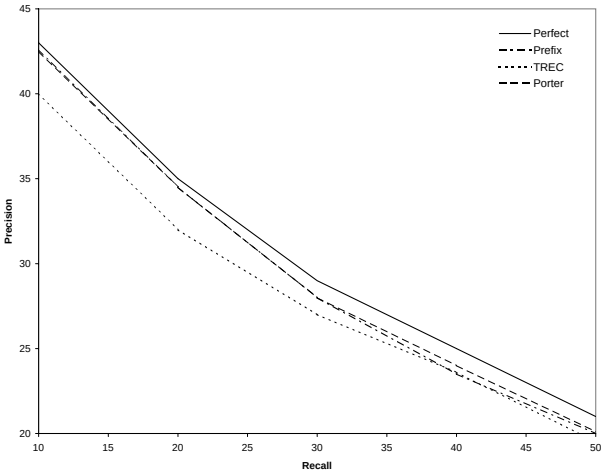

Figure 2: Long queries on the TREC-2 dataset.

From the TREC data set we generated two sets of queries: (i) short queries which were titles of the documents (up to 10 words long), and (ii) long queries, which contained not only the title, but also the domain and narrative section of the query (which were often between 50 and 200 words long). Stop words were removed from the queries. Ranking of retrieved documents was computed using TFIDF.[4] Figures 1 and 2 show the results for the short and long queries respectively. As expected, precision is higher for long queries (because of the higher expected overlap between the query and the document; several of the short queries did not have any terms in common with the relevant files). Stemming does not seem to have as much effect with longer queries as it does with shorter queries, though in both cases, the stemmers seem to help.

We ran similar experiments on some other data sets. For

[2] Word frequencies were based on the frequencies in a year's worth of recent Associated Press and Reuter's news wire articles.

[3] As discussed by [3], it is difficult to evaluate IR performance by experimenting with either a small data set or a small number of queries, since neither are likely to accurately model the wide variation in query lengths or vocabulary distributions that can occur in the general case. However, we believe that experimental work using a wide variety of data sets and queries can be useful to suggest areas of further exploration.

[4] The top 4500 hits as ranked by TFIDF were retrieved for judging relevance.

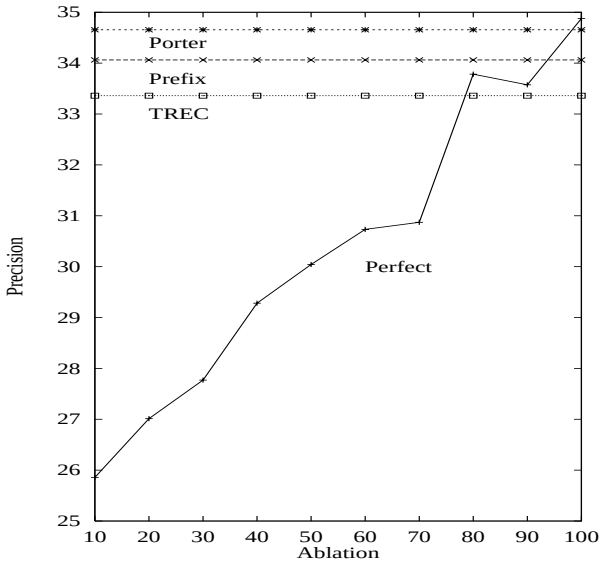

Figure 3: The dict. based stemmer vs. other stemmers.

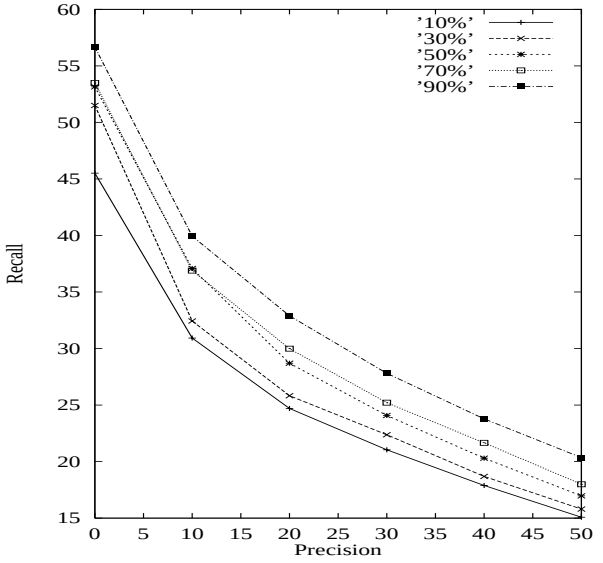

Figure 4: Effect of random voc. ablation on IR perf.

these, we used the Cranfield-1400, MED and ADI data sets. Regarding the length, number of documents and number of queries, the Cranfield, MED and ADI data sets were all similar in sizes to each other, but on very different topics with dissimilar vocabulary. Each of these was about 2000 short documents. The relevance data was for approximately 200 queries, each of length approximately 20 words. The resulting improvements in precision for different values of recall ranged between 1 and 5 percent. Performance on the MED data set (not shown here) demonstrated that some stemming seems to improve performance, with the dictionary based stemmer being much better than the others in the MED dataset.

To evaluate the impact of stemming performance on IR, we re-ran the IR experiments with different levels of coverage and accuracy for the stemmer. While this would be impossible to do with a rule based stemmer, such as Porter, it is easy to do so with our dictionary based "perfect" stemmer by randomly ablating appropriate portions of the underlying equivalence classes. The resulting change in performance is shown in Figure 3 and 4. Figure 3 shows that increases in stemmer coverage yield improvements in IR precision. As can be seen in the graph, there is a strong correlation between stemming performance and IR. This graph is valuable because it shows that the correspondence is not necessarily linear, and that for different points on the precision-recall curve, different levels of stemming accuracy may suffice with TFIDF rankings. This ability, to trade off IR accuracy for smaller stemmers can be very valuable in certain contexts.

4 Conclusions

This poster attempts to study the impact of stemming performance on retrieval. Most previous efforts at evaluating the effectiveness of stemmers for information retrieval experimented with different stemmers, but were unable to correlate changes in stemming accuracy with corresponding changes in IR performance. The main contribution of this poster is the proposal to use a dictionary based stemmer – basically a perfect stemmer, whose performance, in terms of both coverage and accuracy can be selectively changed – to evaluate its effect on retrieval. A stemmer such as this can enable system designers to accurately evaluate the relative trade-offs in improving stemming accuracy – and its concomitant increase in stemmer complexity – with desired levels of IR performance. Our experiments with data sets of varying document and query sizes found that improvements in stemming accuracy yielded corresponding improvements in retrieval precision with short queries. At its best performance level, the dictionary based stemmer yielded significant improvements in precision on short queries. The almost linear improvement in performance for it with increasing coverage suggests that further improvements in TFIDF ranking for IR may be possible by including specialized, domain specific lexicons in the stem-classes.

References

[1] Damerau, F. J. A technique for computer detection and correction of spelling errors. *CACM 7*, 3 (1964), 171–176.

[2] Harman, D. How effective is suffixing? *JASIS 42*, 1 (1991), 7–15.

[3] Hull, D. A. Stemming algorithms: A case study for detailed evaluation. *JASIS 47*, 1 (1996), 70–84.

[4] Kraaij, W. Viewing stemming as recall enhancement. *SIGIR-96* (1996), 40–48.

[5] Krovetz, R. Viewing morphology as an inference process. *SIGIR-93* (1993), 191–202.

[6] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl. 10*, 8 (1966), 707–710.

[7] Lovins, J. B. Development of a stemming algorithm. *Mech. Trans. and Comp. Ling. 11* (1968), 22–31.

[8] Paice, C. D. Method for eval. of stemming algorithms based on error counting. *JASIS 47*, 8 (1996), 632–649.

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