Conflation-based Comparison of Stemming Algorithms

Michael Fuller Justin Zobel
Department of Computer Science, RMIT
GPO 2476V, Melbourne 3001, Australia
{msf,jz}@cs.rmit.edu.au

Abstract

In text database systems, query terms are stemmed to allow them to be conflated with variant forms of the same word. On the one hand, stemming allows the query mechanism to find documents that would otherwise not contain matches to the query terms; on the other hand, automatic stemming is prone to error, and can lead to retrieval of inappropriate documents. In this paper we investigate several stemming algorithms, measuring their ability to correctly conflate terms from a large text collection. We show that stemming is indeed worthwhile, but that each of the stemming algorithms we consider has distinct advantages and disadvantages; choice of stemming algorithm affects the behaviour of the retrieval mechanism.

Keywords: information retrieval, document databases, digital libraries, word disambiguation.

1 Introduction

Queries to text database systems consist of words to be matched to the content of stored text [15]. A query can be Boolean, where the words are connected by Boolean operators and retrieved documents must satisfy the given condition; or ranked, where the documents with the greatest statistical similarity to the query are retrieved as answers. For both kinds of query there must be a mechanism for deciding whether a given query term corresponds to a given word in a document. The simplest correspondence method is to allow exact matching only; for example, "trains" would match itself only, and a document that contained "Trains" but not "trains" would not be recognised as a match. Another simple correspondence method is case-insensitive exact match.

A more general form of correspondence is known as *stemming*, in which query terms and document terms are reduced to a common root word by removal of affixes. Terms match if they have a common root. For example, "trains" might be stemmed to "train", which would match "train", "training", and of course "trains". Each word thus has a set of *conflations*, that is, words that have the same root. Several stemming algorithms have been proposed [8, 9, 10, 11, 13, 17, 18], based on different principles; each produces rather different sets of conflations.

Stemming has usually been measured by its impact on querying: since stemming changes the documents that are retrieved in response to a query, it has the potential to change the quality of the set of answers. Despite the fact that stemming—and choice of stemmer—can dramatically change the number of documents that match a query term, the results of information retrieval experiments have been mixed, ranging from negligible difference in average performance to definite but small improvements [5, 7, 10]. However, measuring average performance can mask significant changes in individual queries, and the metrics used for evaluating performance can allow the effects of stemming to seem small, even if the documents sets retrieved are changed significantly.

Instead of using variations in recall and precision as an indirect measure of stemming efficacy, it is possible to evaluate stemming algorithms by the accuracy of the conflations they produce. This was the approach of Paice [12] and of the work presented in this paper: to evaluate stemming, not by its impact on query evaluation, but by direct consideration of the set of terms that are determined by stemming to be equivalent. We investigate three well-known stemming methods, the S stemmer, Lovins's algorithm, and Porter's algorithm, and compare them to a new dictionary-based method that we have developed. Most existing algorithms use word-formation rules and a small lexicon of common prefixes and suffixes to reduce words to probable roots; our method is based on a large public-domain dictionary of words and affixes.

We show experimentally that these methods have rather different behaviour. Using the TREC

Proceedings of the Third Australian Document Computing Symposium, Sydney, Australia, August 21, 1998.

text database and queries, we show that Lovins's method and Porter's method are fairly error-prone, while the dictionary method does not discover as many correct conflations. However, our results indicate that for typical text retrieval applications stemming should be used, and that suffix removal with the Porter algorithm is unlikely to be significantly bettered.

2 Stemming and document retrieval

The task of query evaluation is to find the documents that match a query. For Boolean querying, matching is defined as a logical condition composed of query terms and Boolean connectives. For ranked querying, each document in the collection is scored against the query, and the documents with the highest scores are fetched as matches. As these querying methods only approximate the user's actual information need, they are measured by their ability to fetch relevant documents, that is, documents that a user has manually judged to be relevant.

Two metrics are commonly used to measure retrieval systems: recall, the proportion of the set of relevant documents that are retrieved; and precision, the proportion of the retrieved documents that are relevant. These are often combined into a single recall-precision figure, or effectiveness, by averaging precision at different levels of recall. Systems are tested by applying them to a standard test collection (consisting of documents, queries, and manual relevance assessments for each query) and computing an average effectiveness across all queries. A great many different scoring functions for ranking have been proposed [20], and, as initiatives such as the TREC experiments [6] have shown, different systems can be of similar effectiveness, yet retrieve very different sets of answers [19].

Most previous investigations of stemming have focused on the impact on effectiveness of changing the stemming mechanism [5, 7, 9, 10, 18]. Harman [5] evaluated the performance of the Lovins, Porter, and S stemmers on retrieval from the small Medlars, Cranfield, and CACM collections. Her conclusions were overall weakly positive: stemming produces at best a non-significant improvement. She tried three techniques to improve performance: using lower weights for expanded terms; only using stemming to broaden queries if the number of query terms was less than ten; and only stemming low-noise terms. None of these resulted in improved retrieval. Her results indicated that whilst stemming does improve the performance of some queries, it weakens that of others. In contrast to Harman's conclusions, Krovetz reported generally positive results from the use of stemming [10]; these experiments used a different set of test collections to Harman's work. However, results were not consistent across all collections.

Interestingly, evidence from other fields is also mixed. For example, Riloff reported that stemming had a deleterious effect on term classification [14]; whereas Sanderson's work on sense disambiguation [16] recorded that random ambiguation of terms had minimal impact on retrieval and that only very accurate re-disambiguation regained lost performance, suggesting that stemming should at worst have no effect and could potentially result in significant gains.

Intuitively these results are quite surprising. For example, the scoring functions used in ranking consider the frequency of the query terms, and give considerably more weight to terms that are relatively rare. Thus if a rare term becomes conflated with a common term, not only are many more documents allocated a score but the term's weight in each document is also greatly reduced. The true behaviour may be even more complex than this, however: the work of Church suggests that the correlation between terms (and therefore, from a retrieval perspective, the usefulness of conflation) can be observed to depend on both their frequency and their distribution [2].

However, assuming that the conflation is accurate and does indeed introduce additional relevant documents, with the standard measures of effectiveness these behaviours tend to cancel out. Such conflations increase recall but degrade precision, with small net impact on effectiveness. Moreover, the design of some test collections can obscure the effect of stemming: the large number of terms in long queries ameliorates stemming errors, and long relevant documents are likely to contain many of the conflations of each query term, reducing the need for stemming. These effects could well be the cause of the contrasting results of Harman [5] and Krovetz [10].

Nonetheless, there are clearly cases in which stemming is helpful, such as when one form of a word is mapped to another; and cases in which erroneous stemming is unhelpful, such as when a word is incorrectly mapped to a common word. In practice, perhaps only a small sample of queries would be significantly affected by changes in stemming; measurement of the effect of stemming on information retrieval should therefore investigate per-query changes rather than overall impact.

Paice assessed stemming performance against predefined concept groups in samples of words taken from the small CISI test collection and two other small sources [12]. Treating stemming as the process of creating concept groups of related terms, a similar approach to that taken in this paper, he evaluated the performance of the Lovins, Porter, and the Paice-Husk stemmer. Paice limited his work to reporting on ideal stemming performance, choosing not to consider term frequency in his evaluation. His results confirm that Lovins is less accurate but more aggressive than Porter, and that the Paice-Husk stemmer, as well as being more accurate, is even more aggressive than Lovins.

3 Stemming algorithms

Stemming algorithms proceed by extracting each word from the text, and, considering that word in isolation, reducing it to a probable root word. Current stemming algorithms do not, for example, use context information to determine the correct sense of each word, and indeed such an approach is unlikely to be helpful. While knowledge of the sense of a word can help guide a stemmer—for example, in the context of income the term "living" should not be stemmed, but in the context of biology it should be stemmed to "live"—such cases are fairly rare. Moreover, given that for most words the vast majority of uses are of one sense only, errors due to inaccurate word-sense analysis are likely to overwhelm any gains that might be made by increased accuracy in stemming.

We now consider several stemming methods in detail, including existing methods and our proposed dictionary technique.

Elementary methods. The simplest form of term conflation is case-folding, thus allowing case-insensitive match. Case-folding is likely to be helpful in most cases, but it is worth noting that this elementary approach can cause errors, for example when the title "Major" is conflated with "major", or the acronym "AIDS" with the term "aids". That is, for even the simplest stemming technique there is potential for effectiveness to degrade for some queries.

Another simple method is the S stemmer [5], in which only a few common word endings are removed: "ies", "es", and "s" (with exceptions). Some practical systems use the S stemmer as, although it does not discover many conflations, it is conservative and rarely produces conflations that surprise the user.

Porter's method. The Porter stemmer [13] removes about 60 suffixes in a multi-step approach, successively removing short suffixes without exceptions. Each step results in the removal of a suffix or the transformation of the root.

Lovins's method. The Lovins stemmer [11] uses a longest match algorithm and exception list to remove over 260 different suffixes. It is the most aggressive of the three algorithmic approaches to stemming.

We used public-domain implementations of the S stemmer, the Porter stemmer, and the Lovins stemmer for the experiments described below.

The dictionary method. Stemming based on affix removal is straightforward to implement and requires minimal run-time resources, but has obvious disadvantages. In particular, a stemming operation that is valid in one case is invalid in another—it is valid to conflate "create" and "creation" but not "state" and "station".

To address this problem we have developed a dictionary-based stemmer. Some dictionaries hold affix information, showing the forms that can be derived from a root word. The public-domain spell checking utility `ispell` includes such a dictionary, with, for each entry, a list of legal affixes. To create a list of "legal" conflations, each affix rule was applied to its root to produce a dictionary of conflation–base term pairs. Where a base term was identical to the conflation of another base term, it was replaced by that conflation's base term. The expanded dictionary contains 41,709 distinct terms.

The dictionary-based approach does have some limitations. The `ispell` dictionary was developed for spelling checking, and includes some derivations that correspond to legal spelling but not the same word; for example, it derives the valid English word "stationer" by appending the affix "er" to the unrelated word "station". However, the number of such errors appears to be fairly small, and as we show below it allows much more accurate conflation than do the other stemming methods. Other problems are that, despite the size of the dictionary, some words are not present; for others, the spelling is given explicitly instead of as a derivation from a root, so that we cannot detect the conflation; and for others, directly related word forms have separate entries to allow different derivations. These problems could be addressed in a production system with a dictionary designed for stemming.

Other techniques. In addition to testing Porter and a Porter variant, Krovetz introduced a new dictionary-moderated inflectional stemmer and a new dictionary-moderated derivational stemmer (now known as KSTEM) [10]. The latter technique stemmed by removing derivational endings, with the requirement that resultant terms must themselves be present in the dictionary. Their experiments showed that this technique, which in principle is similar to Porter with restrictions, yielded small but significant improvements in retrieval effectiveness.

Xu and Croft introduced a novel technique that combines the use of an aggressive initial stemmer (using trigram conflation or Porter) with term co-occurence data to create corpus-specific equivalence classes; they reported similar or improved retrieval performance compared with that when using KSTEM or Porter stemming [18]. Interestingly, this technique was directly applied to a Spanish corpus with a similar outcome, and could potentially be applied to corpora in other languages without requiring significant additional linguistic knowledge.

Last, Jacquemin has described a truncation-based technique for conflating similar phrases [8].

4 Test data

We chose to test the stemmers by examining the conflation sets they produce for a real text database. The database used was the second disk of TREC data [6], which is one gigabyte of text with a vocabulary of 363,553 distinct terms after case folding. Rather than examine the conflations of all terms, we restricted our attention to terms occurring in TREC queries, both to limit the resources required to complete this work and to obtain a realistic view of the effect of stemming: many of the terms occurring in the TREC data would never be used in a query. The queries used were numbers 211–235 and 251–300; after removal of stopwords (common words such as "the" and closed-class words such as "furthermore"), there were 1,093 distinct terms in total.

Using the TREC vocabulary, we applied each stemmer—the S stemmer, Lovins's, and Porter's stemmers, and our dictionary stemmer—to determine a conflation set for each of the query terms. These were then pooled, giving on average 13.5 conflations per term. We then manually judged the conflations, using the criterion that a conflation was correct if it and the query term could reasonably have been derived from the same root word. Overall, there was an average of 7.9 correct conflations per query term. Some of these "correct" conflations may not be appropriate in the context of particular query or document; our judgements were on the basis of the information that would be available to a stemmer, namely the word alone.

For some terms there were only a few conflations. At the other extreme, each of the terms "care", "cure", and "current" had 56 distinct case-folded conflations; those for "cure" are shown in Figure 1, of which 10 were judged correct. Interestingly, many of the false conflations are a consequence of typographic errors.

Stemmer	Attempted	Correct	Missed
No stemming	1.0	1.0	6.9
Perfect	7.9	7.9	0.0
S	1.7	1.7	6.1
Dictionary	4.1	3.8	4.1
Porter	6.3	5.2	2.6
Lovins	11.1	5.9	2.0

Table 1: *Performance of each stemmer, as measured by the number of conflations attempted, the number of these that are correct, and the number of correct conflations missed, averaged over all 1,093 query terms.*

Our manual judgements in effect provide us with an additional, *perfect* stemmer that finds all, and nothing but, correct conflations for the given query terms. Our evaluation of the perfect stemmer and the other stemmers is discussed in the next section.

5 Experiments and results

Our first experiment was a comparison of the impact of each stemmer on retrieval effectiveness. This experiment was inconclusive, with the different stemmers yielding small overall differences in performance, in line with earlier work [5, 10]. However, as Harman also noticed [5], the effect on performance varied from query to query. As a consequence of the results described below we will further explore the impact on effectiveness in ongoing work.

The second experiment was to directly measure the performance of the stemmers by examining their ability to find the correct conflations. There are two dimensions to these results: how many conflations were attempted for each query term, and how many were correct. Results are shown in Table 1; the "no stemming" data shows behaviour if stemming is not performed. As can be seen, failure to stem means that many reasonable conflations are overlooked, and the S stemmer, while highly reliable, only finds 0.7 additional terms per query term, just 22% of those possible. On the other hand, the Lovins stemmer is highly aggressive, but (in comparison to Porter) the 4.8 additional conflations per query term only contain 0.7 additional correct terms; Lovins has an accuracy of 53% while Porter has an accuracy of 83%. The dictionary stemmer is even more accurate, at 93%, but misses many terms found by the Porter and Lovins methods; these three stemmers found 48%, 67%, and 75% of possible conflations, respectively.

cur cura **curable** curate curated curates curation **curative curatives** curator **curators cure cured** curency curently **curer cures** curi curia curial curies **curing** curious curiously curity curium curless curly curr currant currants currator curre curren currencies currency currenly current currently currentness curried currier curries curro curry currying curs curse cursed curses cursing cursive curso cursor cury **uncured**

Figure 1: *Distinct conflations for the term "cure" due to the four stemmers. Conflations that were judged correct are shown in bold.*

Note that, since pooling was used to derive the set of possible conflations, the recall figure (proportion of the correct conflations found by each stemmer) may be inflated; other stemmers might find further correct conflations. There is a surprising lack of overlap between the conflations found by each stemmer.

Interestingly, visual inspection of the errors—that is, the incorrect conflations and the correct conflations that were missed—suggested that although the errors were numerous they might be unimportant, since many of them corresponded to rare or unusual forms of words. We thus conducted a third experiment, where we used the collection-wide statistics to count the number of word occurrences that conflated to each query term. For example, the term "train" occurs 4,890 times in the TREC data, "training" occurs 17,808 times, and "trains" occurs 1,368 times, giving a total of 24,066 occurrences for those conflations of "train".

Results are shown in Table 2. As can be seen, a rather different picture emerges. The S stemmer finds about 61% of the correct conflations by frequency, up from the 22% of our first experiment. Similar increases can be observed for each of the other stemmers, with the dictionary stemmer jumping to 84%, Porter 89%, and Lovins 90% of the correct conflations by term frequency. The accuracy of Lovins rises to 86%, while Porter is 97% and the dictionary stemmer is around 94%.

A distinctly new picture has emerged through consideration of term frequency. Even the simplistic S stemmer is surprisingly effective and the coverage of Porter is as good as the more aggressive Lovins. Accuracy is markedly better for Porter and Lovins, to the extent that the dictionary stemmer, by this measure, can be seen to be *less* reliable than Porter. Indeed, Porter's performance leaves little scope for practical improvement.

6 Conclusions

We have compared the performance of several stemmers on query terms, measured by their ability to find correct conflations and to find nothing but correct conflations, weighted by the in-collection frequency of the terms involved. The success of the Porter stemmer in achieving 97% accuracy at 90% coverage of potential conflations strongly suggests that stemming is worthwhile. Why previous research has been inconclusive is not clear; possible explanations include that stemming may introduce wrong senses of words, that it adversely affects term weighting, or that the recall-precision metric is in this case a poor method of measurement.

Stemmer	Attempted	Correct	Missed
No stemming	15,331	15,331	21,290
Perfect	36,621	36,621	0
S	22,492	22,470	14,151
Dictionary	32,540	30,649	5,972
Porter	33,584	32,652	3,969
Lovins	38,680	33,118	3,503

Table 2: *Performance of each stemmer, as measured by the count of word occurrences that match each query term. The reported results are the number of conflations attempted, the number of these that are correct, and the number of correct conflations missed, averaged over all 1,093 query terms.*

From our results, there seems little reason to use any stemmer other than Porter—it is reliable and finds most of the conflations. As noted above however, the dictionary used was not designed for this purpose, and, in practice, both better reliability and coverage could be expected. The work of Xu and Croft [18] and of Krovetz [10] suggests that truncation techniques moderated by reference to a machine-tractable dictionary could result in improved performance for the simple dictionary stemmer used here.

The next step is to assess the impact of stemming on term weighting. Future work will extend the frequency-based results reported here by using term weighting to examine the impact on similarity measure performance, and explore the use of residual IDF to control selective stemming, as suggested by Church [2].

Acknowledgements

This work was supported by the Australian Research Council.

References

[1] Nicholas J. Belkin, A. Desai Narasimhalu and Peter Willet (editors). *Proceedings of the 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, Pennsylvania, U.S.A., 17–31 July 1997. ACM.

[2] Kennneth Ward Church. One term or two? In Fox et al. [4], pages 310–318.

[3] W. Bruce Croft and C. J. van Rijsbergen (editors). *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin University*, Dublin, Ireland, 3-6 July 1994. Springer-Verlag.

[4] Edward A. Fox, Peter Ingwerson and Raya Fidel (editors). *Proceedings of the 18th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Seattle, Washington, USA, 9-13 July 1995. ACM.

[5] Donna Harman. How effective is suffixing? *Journal of the American Society for Information Science*, Volume 42, Number 1, pages 7–15, 1991.

[6] Donna Harman. Overview of the second text retrieval conference (TREC-2). *Information Processing & Management*, Volume 31, Number 3, pages 271–289, 1995.

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

[8] Christian Jacquemin. Guessing morphology from terms and corpora. In Belkin et al. [1], pages 156–165.

[9] Wessel Kraaij. Viewing stemming as recall enhancement. In Hans-Peter Frei, Donna Harman, Peter Schäuble and Ross Wilkinson (editors), *Proceedings of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 40–48, Zurich, Switzerland, 18-22 August 1996. ACM.

[10] Robert Krovetz. Viewing morphology as an inference process. In Robert Korfhage, Edie Rasmussen and Peter Willett (editors), *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 191–202, Pittsburg, U.S.A., June 27 – July 1 1993. ACM.

[11] J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computation*, Volume 11, Number 1-2, pages 22–31, 1968.

[12] Chris D. Paice. An evaluation method for stemming algorithms. In Croft and van Rijsbergen [3], pages 42–50.

[13] M. Porter. An algorithm for suffix stripping. *Program*, Volume 14, Number 3, pages 130–137, 1980.

[14] Ellen Riloff. Little words can make a big difference for text classification. In Fox et al. [4], pages 130–136.

[15] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA, 1989.

[16] Mark Sanderson. Word sense disambiguation and information retrieval. In Croft and van Rijsbergen [3], pages 142–151.

[17] Evelyne Tzoukermann, Judith L. Klavans and Christian Jacquemin. Effective use of natural language processing techniques for automatic conflation of multi-word terms: the role of derivational morphology, part of speech tagging, and shallow parsing. In Belkin et al. [1], pages 148–155.

[18] Jinxi Xu and W. Bruce Croft. Corpus-based stemming using co-occurence of word variants. *ACM Transactions on Information Systems*, Volume 16, Number 1, pages 61–81, January 1998.

[19] Justin Zobel. How reliable are large-scale information retrieval experiments? In *Proceedings of the 21st International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 308–315, Melbourne, Australia, August 1998.

[20] Justin Zobel and Alistair Moffat. Exploring the similarity space. *SIGIR Forum*, Volume 32, Number 1, pages 18–34, 1998.