Conservative stemming for search and indexing

Marie-Claire Jenkins, Dan Smith
School of Computing Sciences
University Of East-Anglia
Norwich NR4 7TJ
UK

{m.jenkins, dan.smith}@uea.ac.uk

Abstract

In this paper, we describe a stemmer which is designed to stem conservatively to orthographically correct word forms and recognizing words which do not need to be stemmed, such as proper nouns. We compare the performance of our stemmer with several other stemmers and propose further work to make this stemmer more effective for information retrieval, topic detection, and other linguistic applications.

Categories and Subject Descriptors

H.3 INFORMATION STORAGE AND RETRIEVAL: H.3.1 Content Analysis and Indexing – *Linguistic processing.*

General Terms

Stemming, linguistic processing, stemmer comparison

Keywords

Stemmer, spelling, spelling rules

1. Introduction

There are many morphological variants of words used in documents, that stemming algorithms are the logical approach to dealing with information retrieval. The root form of the word can be found and the system is then able to match words and their meaning based on this approach. Stemming algorithms have the advantage of reducing the corpus size thus making information retrieval a faster process. There are a number of stemmers available, notably the Lovins stemmer [1], Paice/Husk stemmer [2] and the Porter stemmer [3]. The problem with these stemmers is often that they tend to be overly aggressive and sometimes reduce words to roots which are non-comprehensible. The stemmer described here is a light stemmer that has been designed to reduce words to roots which are complete words. This is particularly useful when using stemming in topic detection.

2. Stemmer overview

Similarly to other stemmers, UEA-Lite operates on a set of rules which are used as steps. There are two groups of rules: the first to clean the tokens, and the second to alter suffixes.

The first group of rules first avoids a small list of six frequent problem words. An improvement to the stemmer would be to expand this list by adding other problem words which the second rule set cannot deal with. Second, possessive apostrophes are removed and contractions are expanded. All hyphens are removed and tokens containing digits are left untouched. Strings which are all upper case and digits are left untouched unless there is a lower case terminal 's' (i.e. transforming plural forms of acronyms to singular forms).

Proper nouns should not usually be stemmed, except to remove possessives; our implementation will respect PoS tags if they are present. If the text is untagged the stemmer uses the simple heuristic that any capitalized token not preceded by sentence breaking punctuation is a proper noun.

Many texts, particularly scientific papers, contain sequences of digits, single letters, and other non-word tokens. Our implementation ignores tokens containing digits, single-letter tokens, and tokens with embedded punctuation.

The second group of rules contains 139 suffix rules, each testing for a specific type of suffix. The rules are set in a particular order so that the longest suffix applicable is used rather a shorter one which could lead to nonsense words and more words not stemmed entirely to their root form.

2.1 Testing

The suffix rules were developed initially on a collection of 112 documents, mostly scientific papers. Testing has been carried on using a further collection of 201 papers, the Moby common words list [4] and the vocabulary list from Wall Street Journal corpus [XXX]. These results are presented below in Tables 1 and 2..

In calculating the performance of the stemmer on the WSJ and scientific paper sets the result of a change made by the stemmer is counted as correct if it results in a correct word that also denotes the same concept. Changes to incorrectly spelled words have been marked as correct where the mis-spelling does not affect the stemming and which would be correct apart from the mis-spelling. Many words only occur in certain forms, most commonly part participles with an un- prefix (e.g. "unwanted" stems to "unwant"). Changes to these and other words which result in grammatically correct but non-existent words have been marked as wrong.

Table 1. Results on WSJ data

	Tokens	Stemmed	Spelled correctly
Number	49,204	20.24%	85.61%
Frequency	1,173,766	13.01%	91.18%

SIGIR'05, August 15–19, 2005.

Table 2. Results on scientific paper collection

	Tokens	Stemmed	Spelled correctly
Number	44,028	15.04%	89.72%
Frequency	1,189,357	13.91%	93.61%

The Moby common words list contains many tokens which are difficult to classify using the criteria we used for the first two experiments. To overcome the ambiguity in what is a word and what is a correct stemming of it we used a dictionary-based approach. All the stemmed terms were checked against the Microsoft office English (UK) dictionary, as the Moby word list was derived from British English sources.

Table 3. Results on Moby common words

	Tokens	Stemmed	Spelled correctly
Number	74,550	4.9%	84.83%

When comparing the UEA-Lite stemmer to the other stemmers available following the method described in Frakes and Fox [6], the following figures were found:

Table 4. UEA-Lite results on the CAVASSOO papers

	Lovins	Paice/Husk	Porter	UEA-Lite
Mean	1.72	1.98	1.16	1.15
Std. Dev.	1.64	1.92	1.40	0.94
Minimum	0	0	0	0
25th % ile	0	0	0	1
Median	1	2	1	2
75th % ile	3	3	2	3-2
Maximum	10	13	9	6

These figures show clearly that UEA-Lite is a light stemmer. It leaves the majority of words untouched, and is most similar to the Porter stemmer, but differs in that it stems words to correctly spelled roots.

3. Aggressive and conservative stemmers

Aggressive stemmers tends to over-stem the given words, thus leading to a large number of different classes. The words that it produces are heavily conflated, and this leads to many different choices for the system. This could in turn lead to confusion and error, because each word has a very different meaning. A more conservative stemmer produces much fewer classes, so it more probable that the stemmed words will still share the same meaning. The "connected component and optimal partition algorithm" tested by Xu and Croft [7] on the Porter stemmer showed that the expansion factor could be reduced from 4.5 to 2.2 to 2.06 by using this method. They found however that this method produced few improvements on the WSJ corpus but a significant improvement on the WEST corpus. They did however opt to use an n-gram approach rather than simple stemming. This experiment shows that aggressive stemming alone produces far more errors than conservative stemming.

Conservative stemmers such as UEA-Lite which produce less classes allow words to retain their meaning by restricting the number of erroneous stemming results. This method is presented by Krovetz [8], who preceded to modify the Porter stemmer to check the word against a dictionary at every five steps of the Porter Stemmer algorithm. Every time the stemmed word was found in the dictionary, it was retained as a correctly stemmed word. It was found however that this method did not always work sufficiently well. The inflectional stemmer experiment resulted in only a very small improvement, but showed that morphology was still important. It used inflectional morphology as a modification of the porter stemmer. A greater improvement was found using the derivational stemmer, which concerned itself more with meaning using again a dictionary approach. All of these different variants of the porter stemmer show that conservative stemming was able to provide us with a solution which enables stemmed words to have less errors overall.

The choice between an aggressive stemmer and a conservative stemmer clearly depends on the task the stemmer is going to be applied to. When used for document retrieval or similarity measures, aggressive stemmers can suffice as they reduce all words to roughly the same root. When the task involves a more delicate operation such as topic detection, the meaning of the words must be conserved as far as possible.

4. Conclusion and further work

We have developed a stemmer which is designed to conservatively stem suffixes to correctly spelled words. Our results show that it consistently meets these goals in approximately 85% or more of words that it stems. It is available for research use in Perl and Java implementations at http://www.cmp.uea.ac.uk/research/stemmer

We are planning to deploy this stemmer in a number of other projects, where its performance will be measured in operational conditions. Further work is planned to improve its performance with proper nouns.

Acknowledgements

We are grateful to Nathalie Lefevre for her help in accessing the CAVASSOO data.

References

[1] Lovins, J., Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics* 11(1-2), 11-31, 1968

[2] Paice, C., Husk, G., Another Stemmer, *ACM SIGIR Forum* 24(3): 566, 1990

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

[4] Ward, G., The Moby Project., 1996 http://www.dcs.shef.ac.uk/research/ilash/Moby/

[5] Wall Street Journal Corpus, http://www.ldc.upenn.edu

[6] Frakes, W., Fox, C., Strength and similarity of affix removal stemming algorithms. *ACM SIGIR Forum,* 37(1), 2003

[7] J. Xu, W. B. Croft, Corpus-Based Stemming Using Cooccurrence of Word Variants. *ACM Trans. Inf. Syst.* 16(1), 61-81, 1998

[8] Robert Krovetz, Viewing Morphology as an Inference Process, *ACM SIGIR*, 191-202, 1993