An Evaluation Method for Stemming Algorithms

Chris D. Paice

Department of Computing, Lancaster University
Bailrigg, Lancaster LA1 4YR, U.K.

Abstract

The effectiveness of stemming algorithms has usually been measured in terms of their effect on retrieval performance with test collections. This however does not provide any insights which might help in stemmer optimisation. This paper describes a method in which stemming performance is assessed against predefined concept groups in samples of words. This enables various indices of stemming performance and weight to be computed. Results are reported for three stemming algorithms. The validity and usefulness of the approach, and the problems of conceptual grouping, are discussed, and directions for further research are identified.

Introduction

Stemming is a widely-used method of word standardisation designed to allow the matching of morphologically related terms, such as "clusters" and "clustering". The idea is that, in a language like English, a typical word contains a *stem* which refers to some central idea or 'meaning', and that certain affixes have been added to modify the meaning and/or to fit the word for its syntactic role. The purpose of stemming is to strip away the affixes and thus reduce the word to its essence.

In practice, some affixes may alter the meaning of a word so greatly that to remove them would be to discard vital information. In particular, deletion of prefixes is not generally felt to be helpful, except in certain domains such as medicine and chemistry. On the other hand, most suffixes in English are considered to be potentially removable. This paper is concerned with stemming in the restricted sense of suffix removal.

A useful summary of various stemming and conflation algorithms is given in the paper by Lennon *et al.* [1]; further algorithms have been described by Frakes [2] and by Paice [3].

Stemming Errors

There are two particular problems in using stemming for word standardisation. In the first place, pairs of etymologically related words sometimes differ sharply in meaning - for example, consider "author" and "authoritarian". In the second place, the transformations involved in adding and removing suffixes involve numerous irregularities and special cases.

Stemming errors are of two kinds: *understemming errors,* in which words which refer to the same concept are not reduced to the same stem, and *overstemming errors,* in which words are converted to the same stem even though they refer to distinct concepts. In designing a stemming algorithm there is a trade-off between these two kinds of error. A *light stemmer* plays safe in order to avoid overstemming errors, but consequently leaves many understemming errors. A *heavy stemmer* boldly removes all sorts of endings, some of which are decidedly unsafe, and therefore commits many overstemming errors.

There have been several investigations into the effects of stemming on retrieval performance in test collections [1,2,4,5]. In most cases, stemming was found to improve retrieval performance, but not by very much, and there were no consistent differences of performance between different stemmers. Harman was unable to show any consistent benefit over not using stemming at all [6].

We might expect a relationship between the weight of stemming used and the retrieval performance in particular types of search; for instance, it might be supposed that heavy stemming is appropriate when high recall is needed. Lennon *et al.* used the degree of dictionary compression as a measure of the weight, or strength, of each stemmer, but found that this did not correlate clearly with the performance in precision-oriented and recall-oriented searches [1].

Although evaluating stemmers on the basis of their effects on retrieval performance may seem appropriate from an IR viewpoint, it is probably unhelpful in practice, because it gives no insight into the specific causes of errors. Moreover, it ignores the fact that stemmers are not used only in IR systems - for example, they may be used in a natural language interface, or in a frame-instantiation program. IR-based evaluations are irrelevant to such applications.

This paper outlines an approach to stemmer evaluation which is based on detecting and counting the actual under- and overstemming errors committed during stemming of word samples derived from actual texts. This permits the computation of a 'stemming weight' index for each stemmer, as well as indices representing the under- and overstemming error rates and the general accuracy. The method involves manually dividing a sample of words into conceptual groups, and referring the actual stemming performance to these groups.

It may be objected that natural words do not fall into entirely clear-cut semantic groups, and that anyway humans cannot be relied upon to make objective assignments. In fact of course, the very practice of stemming obviously relies on an assumption of semantic grouping, even if it is somewhat approximate. Our assumption here is that careful human judgment can produce groups which are in reasonable agreement with 'reality', and that these can be used for evaluation purposes.

In using a sample of words taken from a natural source, we encounter the question of whether to base the evaluation on all the individual word tokens in the sample, or on only the distinct word types. In the first case, we would determine the frequencies of all the word types and take these into account in computing the performance indices. This is perfectly straightforward, but it is found that the results then obtained tend to be dominated by the way the stemmer handles a quite small number of high frequency word groups - for example, common verbs such as "be"/"being"/"been", "do"/"doing"/"done" etc. This is unfortunate since in practice common and irregular forms are often handled by lexical lookup anyway. Stemmers are mainly required to deal with words of relatively rare and unpredictable occurrence. In view of this, our evaluation method ignores occurrence frequencies, and uses word types rather than tokens. An incidental benefit is that there is no need to remove syntactic function words - nor to worry about what should be included in the stoplist - since it can be shown that inclusion or exclusion of these words has very little effect on the results.

Computation of Performance Indices

Suppose we have a sample of W different words, partitioned into 'concept groups' each containing forms which are both semantically and morphologically related to one another. For each concept group two totals may be computed.

Firstly, since a perfect stemmer should merge every member of a concept group with every other, the number of pairs of different words in the group defines the 'desired merge total' DMT_g for that group. This is given by

$$DMT_g = 0.5\, n_g(n_g - 1)$$

where n_g is the number of words in the group.

Secondly, a perfect stemmer should not merge any member of the present concept group with any word which is not in the group. Thus, for every group there is a 'desired non-merge total' DNT_g given by

$$DNT_g = 0.5 n_g(W - n_g)$$

By summing these two totals over all groups in the word sample, we obtain the *global desired merge total* $GDMT$ and the *global desired non-merge total* $GDNT$ respectively. Each equation contains a 0.5 factor to compensate for double counting of pairs during the summation.

After applying a stemmer to the sample, we are likely to find that some of the groups still contain two or more distinct stems. In such groups there are understemming errors to be counted. Thus, suppose a concept group of size n_g contains s distinct stems after stemming, and that the numbers of instances of these stems are $u_1, u_2, \dots u_s$. The number of understemming errors for the group (the 'unachieved merge total' UMT_g) is given by

$$UMT_g = 0.5 \sum_{i=1..s} u_i(n_g - u_i)$$

Summing this quantity over all groups we obtain the *global unachieved merge total* $GUMT$; the understemming index UI is now given by the ratio $GUMT/GDMT$.

After stemming, we also expect to find cases where the same stem occurs in two or more concept groups. The procedure here is to gather all cases of a particular stem into a 'stem group'; now any stem group whose members are derived from two or more different concept groups contains overstemming errors which need to be counted.

Consider a stem group which contains n_s items which are derived from t different concept groups, and suppose that the numbers of representatives of these concept groups are $v1, v2, \dots vt$. The number of overstemming errors for this stem group is represented by the 'wrongly-merged total' WMTs, given by:

$$WMT_s = 0.5 \sum_{i=1..t} v_i(n_s - v_i) \qquad \{1\}$$

Summing this quantity over all stem groups we obtain the *global wrongly-merged total* $GWMT$; the overstemming index OI is now given by the ratio $GWMT/GDNT$.

It is clear that for a heavy stemmer OI will be rather high and UI low, whereas for a light stemmer the situation will be reversed. The ratio of these two quantities may therefore be taken as a measure of the *stemming weight* SW:

$$SW = OI/UI\,.$$

Because we have no precise theory to account for the relationship between the understemming and overstemming tendencies, we lack any satisfactory framework in which to assess particular UI and OI values. It will commonly occur that one stemmer will be better than another in terms of UI, but worse in terms of OI. So is there any way to judge whether one is better than another overall? And does the question even have meaning? Regarding the second question we may observe that, if the difference in weight is quite small and the difference in UI and/or OI is large, then it probably does make sense to talk about the relative general accuracy of the two stemmers, at least for the word sample under consideration.

In order to obtain some kind of baseline against which the general accuracy of a stemmer may be judged, we refer to the process of *length truncation* - that is, reducing every word to just its first q letters (words shorter than q being left unchanged). Length truncation is the crudest method of stemming, and we would obviously expect any rule-based or table-based stemmer to do better. Note however that length truncation refers to not just one but a series of stemmers, each with a different value of q; for IR purposes truncation lengths of 5, 6 and 7 seem to be the most useful.

The idea here is that if we determine values of UI and OI for a series of truncation lengths, the (UI,OI) coordinates define a *truncation line* against which any stemmer can be assessed. Any reasonable stemmer will give a (UI,OI) point between the truncation line and the origin; in general, the further away the point is from the truncation line, the better the stemmer can be said to be. Specifically, a performance measure which we may call the *error rate relative to truncation*, or $ERRT$, can be obtained by extending a line from the origin **O** through the (UI,OI) point **P** until it intersects the truncation line at **T**, as illustrated in Figure 1. $ERRT$ is then simply defined as

$$ERRT = \text{length}(\mathbf{OP})/\text{length}(\mathbf{OT})\,.$$

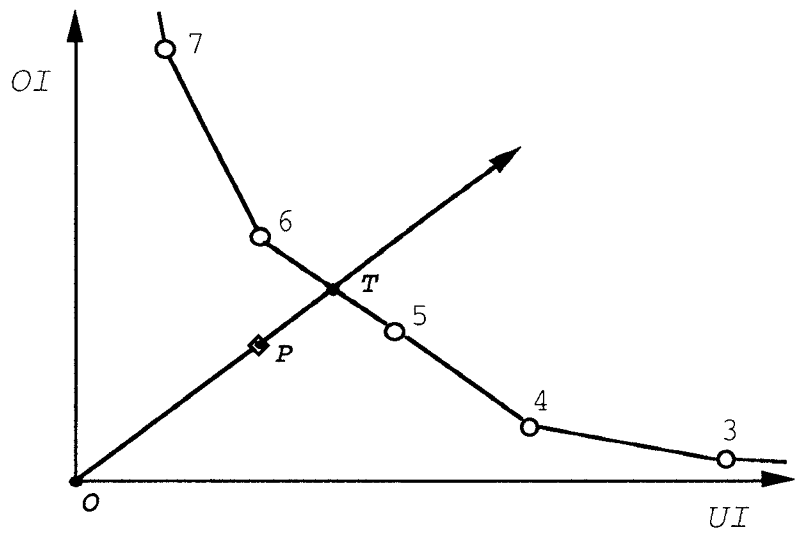

Figure 1: computation of ERRT value

Experimental Arrangements

A suite of computer programs was written to permit computation of the above indices. The processing falls broadly into four parts:

-- conversion of a source text sample into a grouped file;

-- computation of a truncation line for the grouped file;

-- application of one or more stemming algorithms to the words in the grouped file, to obtain values of *UI*,*OI* and *SW*;

-- computation of the *ERRT* value for each stemmer by comparing its (*UI*,*OI*) point to the truncation line.

The only stage where human effort is involved is in the construction of the groups. The grouping program presents to the user an alphabetic display of all the distinct words from the source. The program makes all the obvious decisions on the user's behalf, but for all uncertain cases refers to the user for the decision. Adjacent groups are separated by inserting 'barriers' into the file.

The grouped file thus produced is not likely to be correct. For one thing, some of the 'obvious' decisions taken by the program may in fact be wrong. For another, alphabetic ordering may sometimes split up certain conceptual groups (consider "read", "readily", "reading", "readjust", "reads"). Thirdly, the user may require time to reconsider some of the more difficult groupings. Consequently, a second and perhaps even third scan of the file needs to be performed, using a standard editor, before the grouping will be satisfactory. The grouping is thus a rather laborious process, though once it is finished the grouped file represents a permanent resource for future use.

A question arises over what to do about irregular verbs. It seems natural and proper that "fly" and "flew" should be placed in the same group, but what about "is" and "were", and "go" and "went"? Stemming relies on morphological regularities and similarities, so it seems wrong to penalise it for not merging totally dissimilar forms. In the event, a rule-of-thumb was used that words should only be grouped together if at least the first two letters were the same. This was partly a matter of convenience, since it is awkward to bring together words which are far apart in the alphabetic list. The rule obviously leads to anomalies - e.g., "bring" and "brought" are grouped, but "buy" and "bought" are not - but because the indices are based on word types rather than tokens it has only a tiny effect on the values of the indices.

During the grouping process, the user is presented with individual, isolated words; no context is given, and so there is no chance to allow for the different meanings of ambiguous words. In making a grouping decision, the user is in effect deciding whether two words *typically* refer to the same underlying concept, given a knowledge of the general domain of the source material.

This still leaves the question whether two words which refer to related but not quite identical concepts should be counted as equivalent. It may be that taking a 'strict' view of semantic equivalence will give materially different results than taking a 'loose' view. To investigate this point, groups were actually defined at two levels of 'tightness', using two kinds of inter-group barrier. First, there is a level of possibly large, loose groups, containing words which may be quite weakly related to one another. Secondly, any loose group may be subdivided into two or more tight subgroups, each containing words which refer to more-or-less identical concepts. This approach means that for each stemmer which is evaluated against a given word sample, two separate sets of performance indices are generated.

Some examples of the two-level grouping are shown in Figure 2. The words abstract and addition have been placed in individual subgroups because they are both ambiguous. Thus, one sense of abstract is related to abstractness, the other to literature abstracts. During the grouping, abstracting was assigned to the latter subgroup because the domain of the source was known to be library science.

(abstract)   (abstraction, abstractly)
(abstracts, abstracting, abstracted, abstractors)

(addition)   (additional, additionally)
(add, adds, adding, added, additive)

(alter, alters, altered, alterations)
(alternate, alternately, alternating, alternations)
(alternative, alternatives, alternatively)

(appropriate, appropriately)   (appropriations)

(authur, author's, authors, authorship)

(authoritarian)   (authoritative)   (authority, authorities)
(authorized, authorization)

(cost, costing, costed, costs)   (costly)

(device, devices)   (devise, devising, devised)

(element, elements, elemental)   (elementary)

(explicate, explicates, explicated, explication, explications)
(explicit, explicitly)

(frame, frames, framing, framed)   (framework, frameworks)

Figure 2 : examples of two-level concept groups
Words enclosed within parentheses are grouped tightly together.
A horizontal line is a major barrier between adjacent loose groups.
(All are actual examples taken from CISI source.)

Performance evaluations were carried out for three stemmers whose details (including specific rule tables) are fully described in the IR literature: the Lovins stemmer [7], the Porter stemmer [8] and the Paice/Husk stemmer [3]. To provide a baseline for computing values of *ERRT*, *UI* and *OI* values were also obtained for simple truncation using truncation lengths of 4, 5, 6, 7 and 8.

A sample of words was obtained by processing all of the titles and abstracts in the CISI test collection, which is concerned with Library and Information Science. This source contained a total of 184,659 words, reduced to 9,757 after deletion of duplicates. Runs were also carried out using two smaller word samples: 1,527 distinct words (derived from 8,947 source words) from a textbook excerpt concerned with computer storage devices, and 3,559 distinct words (from 32,098 source words) from the texts of 14 papers on agriculture.

In order to investigate the influence of sample size on the performance indices, four subsamples were prepared from the CISI text source, by taking every nth line from complete collection, with n values of 2, 4, 8 and 16, and then preparing grouped files as usual. The resulting word samples contained 7,304, 5,395, 3804 and 2,654 word types respectively, compared with 9,757 for the full CISI vocabulary.

Results

Although the values of the indices were markedly different for the different word samples, the *patterns* of values were much the same in all cases, so that similar conclusions could be drawn for each sample. Table 1 shows the results obtained for the CISI sample. If we compare the corresponding values obtained for the and loose levels of grouping, we find that all four indices vary in the ways we would expect. Understemming will obviously be greater, and overstemming less, for loose than for tight grouping, and this in turn causes considerably smaller SW values for the loose grouping level. The $ERRT$ values are larger with loose grouping; this too is understandable since loose semantic relationships are naturally reflected in weaker and less systematic morphological similarities.

	---- *tight grouping* ----				---- *loose grouping* ----			
	UI	*OI*	*SW*	*ERRT*	*UI*	*OI*	*SW*	*ERRT*
trunc(4)	0.062	0.000814	0.013127	------	0.099	0.000706	0.007155	------
trunc(5)	0.176	0.000262	0.001487	------	0.258	0.000183	0.000710	------
trunc(6)	0.337	0.000073	0.000218	------	0.442	0.000022	0.000050	------
trunc(7)	0.527	0.000028	0.000054	------	0.633	0.000002	0.000004	------
trunc(8)	0.700	0.000012	0.000017	------	0.780	0.000000	0.000000	------
Lovins	0.326	0.000063	0.000193	0.92	0.459	0.000020	0.000044	1.00
Paice/Husk	0.121	0.000118	0.000978	0.55	0.257	0.000051	0.000197	0.67
Porter	0.374	0.000028	0.000074	0.76	0.542	0.000004	0.000007	0.88

Table 1 : Stemming performance indices for the CISI word sample

If now we compare the *pattern of values* within the tight and loose sections of Table 1, we can find no marked differences, and this suggests that the properties and validity of the indices are not strongly affected by the level of grouping - provided presumably that a consistent strategy is used.

Comparing the three stemmers with one another, the relative values of the four indices may be summarised as follows:

$$UI(\text{Porter}) > UI(\text{Lovins}) > UI(\text{Paice/Husk})$$

$$OI(\text{Paice/Husk}) > OI(\text{Lovins}) > OI(\text{Porter})$$

$$SW(\text{Paice/Husk}) > SW(\text{Lovins}) > SW(\text{Porter})$$

$$ERRT(\text{Lovins}) > ERRT(\text{Porter}) > ERRT(\text{Paice/Husk})$$

Although the magnitudes of the differences varied a good deal, and in a couple of cases were only marginal, the above inequalities actually held for all of the word samples tested. In terms of the UI index, Lovins was noticeably closer to Porter than to Paice/Husk.

If we take *ERRT* as a general indicator of performance accuracy, we would have to conclude that Paice/Husk is a better stemmer than Porter, which is in turn better than Lovins. However, the differences in stemming weight between Paice/Husk and Porter are so great that it is probably meaningless to compare their accuracy: Paice/Husk is a heavy stemmer and Porter a light stemmer, and presumably each is suited to a different task.

It is helpful to look at Figure 3, which plots *UI* vs. *OI* for the CISI sample at the tight grouping level. This not only highlights the great difference in weight between Paice/Husk and Porter, it also casts light on the performance of Lovins. Notice first that the truncation line is convex towards the origin; this is as expected, given the generally inverse relationship between between *UI* and *OI*. However, a line joining the performance points for Paice/Husk, Lovins and Porter is clearly concave towards the origin, suggesting that Lovins is genuinely less accurate than either of the other stemmers. This relationship holds for both tight and loose levels of grouping for all the word samples tested .

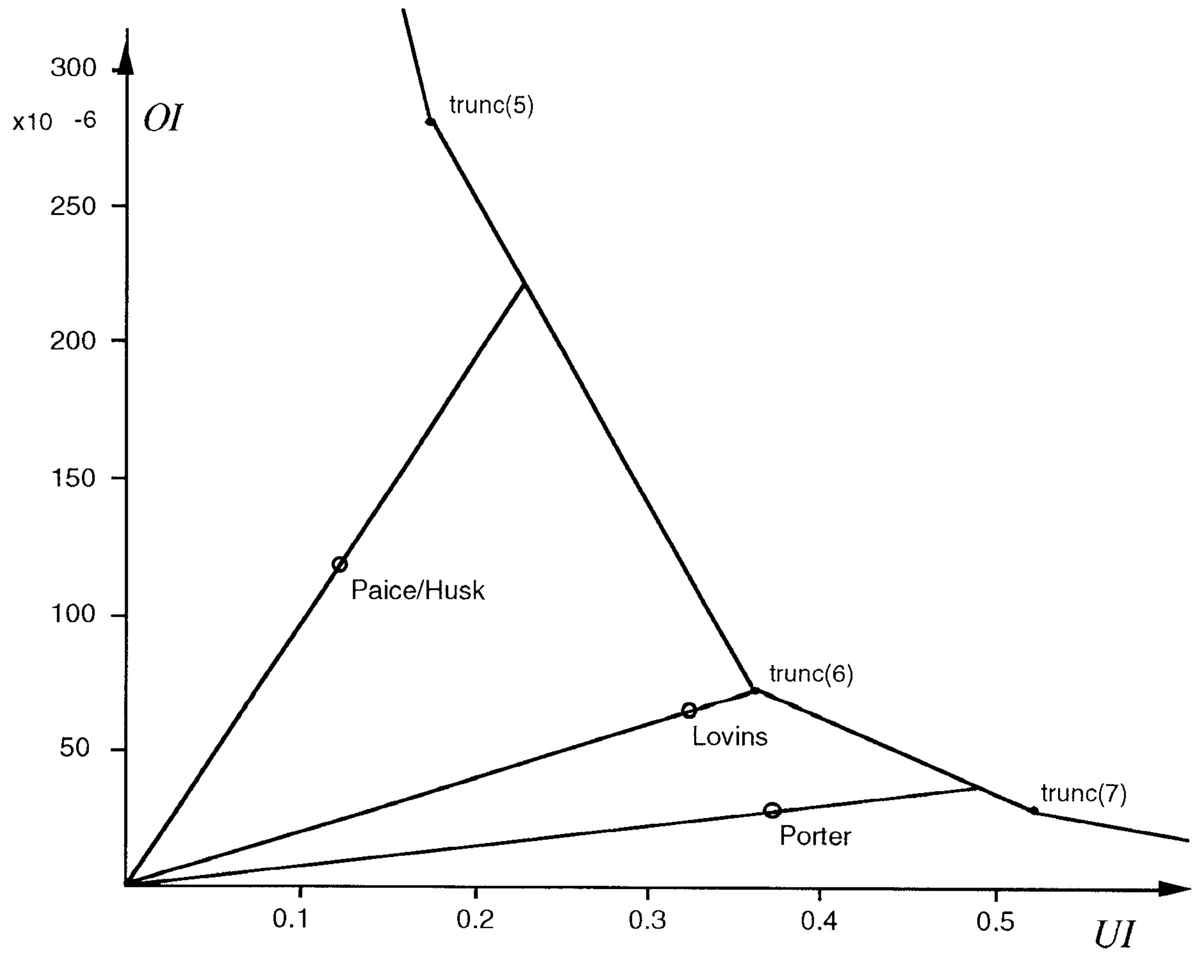

Figure 3 : UI x OI plot for CISI sample (tight grouping)

Lennon *et al.* represented the weights of their stemmers by the dictionary compression each could achieve. Their results for Lovins and Porter are compared with ours in Table 2. Our results from the CISI sample are closest to theirs from the Brown linguistic corpus; oddly, this appeared to be the least similar source to ours. However, our ratio for the compression by Lovins compared to

Porter was 1.14, whilst theirs were all in the range 1.13 to 1.18. Our compression values confirm clearly that Porter is a lighter stemmer than Lovins, and also that Paice/Husk is much heavier.

	--- *this work* ---		--- *Lennon et al. 1981* ---			
	n	CISI	Brown	NPL	Inspec	Cranfield
sample words	9,757	100.0	---	---	---	---
tight groups	5,101	47.7	---	---	---	---
loose groups	4,350	55.4	---	---	---	---
Lovins stems	5,409	44.6	45.8	39.2	39.5	30.9
Paice/Husk stems	4,755	51.3	---	---	---	---
Porter stems	5,964	38.9	38.8	34.6	33.8	26.2

Table 2 : Dictionary compression by three stemmers.
n represents numbers of items;
other values are percentage compression rates.

We now turn to the effect of sample size. It appears that *UI*, representing the internal structure of the concept groups, is fairly insensitive to sample size. *OI* however (and consequently *SW*) falls off sharply as sample size increases. *ERRT* shows a less consistent tendency: the values for Lovins and Porter reach a minimum at about 5,000 sample words, whereas Paice/Husk decreases throughout, but shows signs of levelling off around 10,000 words.

We may recall that *OI* is defined as *GWMT/GDNT*. The global desired non-merge total *GDNT* can be expected to depend on the square of the word sample size *W*. We may therefore infer that the global wrongly-merged total *GWMT* increases less fast, and this in turn implies that the average value of formula {1} is less than proportional to *W*. This can be explained if the average stem-group size shows little or no tendency to increase with *W*; this is plausible since it is likely that as the source grows, the fresh introductions will increasingly tend to be non-domain-related singleton words.

Findings

The general results of our experiments may be summarised as follows:

-- The specific values of the performance indices vary markedly depending on the source text.

-- For a particular word source, the values of *UI* are fairly insensitive to sample size, whereas values of *OI* and *SW* fall off sharply as sample size increases. *ERRT* values show modest fluctuations.

-- In terms of the stemming weight *SW*, Porter is a light stemmer and Paice/Husk a heavy stemmer.

-- The difference in the *SW* value between Porter and Paice/Husk is so great that it is probably meaningless to compare their performance.

-- The Lovins stemmer seems to be generally less accurate than either of the other two stemmers.

-- The choice of grouping level does not appear to be a critical matter, provided that a consistent strategy is used in each case.

The lightness of the Porter stemmer is in agreement with earlier findings that Porter is significantly lighter than Lovins and several other algorithms [1].

Final Comments

The author is well aware of various doubts and problems with the methods described in this paper. Further work is clearly needed to explore the validity of the approach and to make the programs more useful. One area of difficulty concerns the subjective and fuzzy nature of the grouping operations, and it would be valuable to have some objective evidence to assist in this activity. Rather than basing the grouping on a display of isolated words it might be better if grammatically tagged words could be presented instead. Part-of-speech tags might be of some use, but semantic tags would appear to be of greater value [9]. Use of tagged text would of course mean that a particular word type might be assigned to different groups on different occasions.

Use of these stemmer evaluation tools (whether in the existing or an enhanced form) is not limited to comparing the performance of existing off-the-shelf stemmers: they can also be used for optimising the rule-tables which the stemmers use. The types of changes considered would of course depend on the nature of the stemmer in question. For example, we might try to make Porter's algorithm stem a little more heavily by adding additional rules or by modifying or relaxing some of the contextual constraints.

With Paice/Husk, the emphasis would be on reducing the number of overstemming errors by retracting or modifying some of the more troublesome rules. In this case, it would be desirable to discover which rules are a serious cause of overstemming errors. This could be done by modifying the stemmer to keep a note of each removed ending (or the rule used to remove it) so that this information is available when the under- and overstemming errors are being counted. It should then be possible to compute individual error indices for each different ending.

It would be interesting to apply this evaluation method to dictionary-based conflation in order to investigate the extent and significance of the performance gap between a well-optimised stemmer and a full dictionary operation.

Acknowledgements. I should like to thank Gareth Husk, Chris Danson and Helen Simpson for their invaluable work in developing various parts of the software.

References

1. Lennon, M., Pierce, D.S., Tarry, B.D., Willett, P. An evaluation of some conflation algorithms for information retrieval. *Journal of Information Science* 1981; 3, 177-183.

2. Frakes, W.B. *Term Conflation for Information Retrieval.* Ph.D. thesis, Syracuse University, NY, 1982.

3. Paice, C.D. Another stemmer. *SIGIR Forum* 1990; **24**, 56-61.

4. Hafer, M.A. and Weiss, S.F. Word segmentation by letter successor varieties. *Information Storage and Retrieval* 1974; ***10***, 371-385.

5. Landauer, C. and Mah, C. Message extraction through estimation of relevance. In: R.N.Oddy *et al.* (Eds.), *Information Retrieval Research.* London: Butterworths, London, 1981, pp.117-138.

6. Harman, D. How effective is suffixing? *Journal of the American Society for Information Science* 1991; *42*, 7-15.

7. Lovins, J.B. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics* 1968; **11**, 22-31.

8. Porter, M.F. An algorithm for suffix stripping. *Program* 1980; **14**, 130-137.

9. Wilson, A. and Rayson, P. The automatic content analysis of spoken discourse: a report on work in progress. In: Souter, C. and Atwell, A., *Corpus-based Computational Linguistics.* Rodopi, Amsterdam & Atlanta GA, 1993.