Snowball: A language for stemming algorithms

Links

Snowball main page
Porter stemmer page

M.F. Porter
October 2001

Summary

Algorithmic stemmers continue to have great utility in IR, despite the promise of out-performance by dictionary-based stemmers. Nevertheless, there are few algorithmic descriptions of stemmers, and even when they exist they are liable to misinterpretation. Here we look at the ideas underlying stemming, and on this website define a language, Snowball, in which stemmers can be exactly defined, and from which fast stemmer programs in ANSI C or Java can be generated. A range of stemmers is presented in parallel algorithmic and Snowball form, including the original Porter stemmer for English.

1 Introduction

There are two main reasons for creating Snowball. One is the lack of readily available stemming algorithms for languages other than English. The other is the consciousness of a certain failure on my part in promoting exact implementations of the stemming algorithm described in (Porter 1980), which has come to be called the Porter stemming algorithm. The first point needs some qualification: a great deal of work has been done on stemmers in a wide range of natural languages, both in their development and evaluation, (a complete bibliography cannot be attempted here). But it is rare to see a stemmer laid out in an unambiguous algorithmic form from which encodings in C, Java, Perl etc might easily be made. When exact descriptions are attempted, it is often with approaches to stemming that are relatively simple, for example the Latin stemmer of Schinke (Shinke 1996), or the Slovene stemmer of Popovic (Popovic 1990). A more complex, and therefore more characteristic stemmer is the Kraaij-Pohlmann stemmer for Dutch (Kraaij 1994), which is presented as open source code in ANSI C. To extract an algorithmic description of their stemmer from the source code proves to be quite hard.

The disparity between the Porter stemmer definition and many of its purported implementations is much wider than is generally realised in the IR community. Three problems seem to compound: one is a misunderstanding of the meaning of the original algorithm, another is bugs in the encodings, and a third is the the almost irresistible urge of programmers to add improvements. For example, a Perl script advertised on the Web as an implementation of the Porter algorithm was tested in October 2001, and it was found that 14 percent of words were stemmed incorrectly when given a large sample vocabulary. Most words of English have very simple endings, so this means that it was effectively getting everything wrong. At certain points on the Web are demonstrations of the Porter stemmer. You type some English into a box and the stemmed words are

displayed. These are frequently faulty. (A good test is to type in *agreement*. It should stem to *agreement* — the same word. If it stems to *agreem* there is an error.) Researchers frequently pick up faulty versions of the stemmer and report that they have applied 'Porter stemming', with the result that their experiments are not quite repeatable. Researchers who work on stemming will sometimes give incorrect examples of the behaviour of the Porter stemmer in their published works.

To address all these problems I have tried to develop a rigorous system for defining stemming algorithms. A language, Snowball, has been invented, in which the rules of stemming algorithms can be expressed in a natural way. Snowball is quite small, and can be learned by an experienced programmer in an hour or so. On this website a number of foreign language stemmers is presented (*a*) in Snowball, and (*b*) in a less formal English-language description. (*b*) can be thought of as the program comments for (*a*). A Snowball compiler translates each Snowball definition into (*c*) an equivalent program in ANSI C or Java. Finally (*d*) standard vocabularies of words and their stemmed equivalents are provided for each stemmer. The combination of (*a*), (*b*), (*c*) and (*d*) can be used to pin down the definition of a stemmer exactly, and it is hoped that Snowball itself will be a useful resource in creating stemmers in the future.

2 Some ideas underlying stemming

Work in stemming has produced a number of different approaches, albeit tied together by a number of common assumptions. It is worthwhile looking at some of them to see exactly where Snowball fits into the whole picture.

A point tacitly assumed in almost all of the stemming literature is that stemmers are based upon the written, and not the spoken, form of the language. This is also the assumption here. Historically, grammarians often regarded the written language as the real language and the spoken as a mere derivative form. Almost in reaction, many modern linguists have taken a precisely opposite view (Farmer, 1965 pp 2-3). A more balanced position is that the two languages are distinct though connected, and require separate

treatment. One can in fact imagine parallel stemming algorithms for the spoken language, or rather for the phoneme sequence into which the spoken language is transformed. Stress and intonation could be used as clues for an indexing process in the same way that punctuation and capitalisation are used as clues in the written language. But currently stemmers work on the written language for the good reason that there is so much of it available in machine readable form from which to build our IR systems. Inevitably therefore the stemmers get caught up in accidental details of orthography. In English, removing the ***ing*** from *rotting* should be followed by undoubling the ***tt***, whereas in *rolling* we do not undouble the ***ll***. In French, removing the ***er*** from *ennuyer* should be followed by changing the ***y*** to ***i***, so that the resulting word conflates with *ennui*, and so on.

The idea of stemming is to improve IR performance generally by bringing under one heading variant forms of a word which share a common meaning. Harman (1991) was first to present compelling evidence that it may not do so, when her experiments discovered no significant improvement with the use of stemming. Similarly Lennon (1981) discovered no appreciable difference between different stemmers running on a constant collection. Later work has modified this position however. Krovetz (1995) found significant, although sometimes small, improvements across a range of test collections. What he did discover is that the degree of improvement varies considerably between different collections. These tests were however done on collections in English, and the reasonable assumption of IR researchers has always been that for languages that are more highly inflected than English (and nearly all are), greater improvements will be observed when stemming is applied. My own view is that stemming helps regularise the vocabulary of an IR system, and this leads to advantages that are not easily quantifiable through standard IR experiments. For example, it helps in presenting lists of terms associated with the query back to the IR user in a relevance feedback cycle, which is one of the underlying ideas of the probabilistic model. More will be said on the use of a stemmed vocabulary in section 5.

Stemming is not a concept applicable to all languages. It is not, for example, applicable in Chinese. But to languages of

the Indo-European (*) group (and most of the stemmers on this site are for Indo-European languages), a common pattern of word structure does emerge. Assuming words are written left to right, the stem, or root of a word is on the left, and zero or more suffixes may be added on the right. If the root is modified by this process it will normally be at its right hand end. And also prefixes may be added on the left. So *unhappiness* has a prefix ***un***, a suffix ***ness***, and the ***y*** of *happy* has become ***i*** with the addition of the suffix. Usually, prefixes alter meaning radically, so they are best left in place (German and Dutch ***ge*** is an exception here). But suffixes can, in certain circumstances, be removed. So for example *happy* and *happiness* have closely related meanings, and we may wish to stem both forms to *happy*, or *happi*. Infixes can occur, although rarely: ***ge*** in German and Dutch, and ***zu*** in German.

One can make some distinction between *root* and *stem*. Lovins (1968) sees the root as the stem minus any prefixes. But here we will think of the stem as the residue of the stemming process, and the root as the inner word from which the stemmed word derives, so we think of root to some extent in an etymological way. It must be admitted that when you start thinking hard about these concepts *root*, *stem*, *suffix*, *prefix* ... they turn out to be very difficult indeed to define. Nor do definitions, even if we arrive at them, help us much. After all, suffix stripping is a practical aid in IR, not an exercise in linguistics or etymology. This is especially true of the central concept of *root*. We think of the etymological root of a word as something we can discover with certainty from a dictionary, forgetting that etymology itself is a subject with its own doubts and controversies (Jesperson 1922, Chapter XVI). Indeed, Jesperson goes so far as to say that

> 'It is of course impossible to say how great a proportion of the etymologies given in dictionaries should strictly be classed under each of the following heads: (1) certain, (2) probable, (3) possible, (4) improbable, (5) impossible — but I am afraid the first two classes would be the least numerous.'

Here we will simply assume a common sense understanding of the basic idea of stem and suffix, and hope that this proves sufficient for designing and discussing stemming algorithms.

We can separate suffixes out into three basic classes, which will be called *d*-, *i*- and *a*-suffixes.

An *a*-suffix, or *attached* suffix, is a particle word attached to another word. (In the stemming literature they sometimes get referred to as 'enclitics'.) In Italian, for example, personal pronouns attach to certain verb forms:

mandargli =	mandare + ***gli***	= to send + to him
mandarglielo =	mandare + ***gli*** + ***lo***	= to send + it + to him

a-suffixes appear in Italian and Spanish, and also in Portuguese, although in Portuguese they are separated by hyphen from the preceding word, which makes them easy to eliminate.

An *i*-suffix, or *inflectional* suffix, forms part of the basic grammar of a language, and is applicable to all words of a certain grammatical type, with perhaps a small number of exceptions. In English for example, the past of a verb is formed by adding ***ed***. Certain modifications may be required in the stem:

fit + ***ed*** -> fitted (double ***t***)
love + ***ed*** -> loved (drop the final ***e*** of love)

but otherwise the rule applies in a regular way to all verbs in contemporary English, with about 150 (Palmer, 1965) exceptional forms,

bear beat become begin bend ....
bore beat became began bent

A *d*-suffix, or *derivational* suffix, enables a new word, often with a different grammatical category, or with a different sense, to be built from another word. Whether a *d*-suffix can be attached is discovered not from the rules of grammar, but by referring to a dictionary. So in English, ***ness*** can be added to certain adjectives to form corresponding nouns (*littleness, kindness, foolishness* ...) but not to all adjectives (not for example, to *big, cruel, wise* ...) *d*-suffixes can be used to change meaning, often in rather exotic ways. So in italian ***astro*** means a sham form of something else:

medico + ***astro*** = medicastro = quack doctor
poeta + ***astro*** = poetastro = poetaster

Generally *i*-suffixes follow *d*-suffixes. *i*-suffixes can precede *d*-suffixes, for example *lovingly*, *devotedness*, but such cases are exceptional. To be a little more precise, *d*-suffixes can sometimes be added to participles. *devoted*, used adjectivally, is a participle derived from the verb *devote*, and ***ly*** can be added to turn the adjective into an adverb, or ***ness*** to turn it into a noun. The same feature occurs in other Indo-European languages.

Sometimes it is hard to say whether a suffix is a *d*-suffix or *i*-suffix, the comparative and superlative endings ***er***, ***est*** of English for example.

A *d*-suffix can serve more than one function. In English, for example, ***ly*** standardly turns an adjective into an adverb (*greatly*), but it can also turn a noun into an adjective (*kingly*). In French, ***ement*** also standardly turns an adjective into an adverb (*grandement*), but it can also turn a verb into a noun (*rapprochement*). (Referring to the French stemmer, this double use is ultimately why ***ement*** is tested for being in the *RV* rather than the *R*2 region of the word being stemmed.)

It is quite common for an *i*-suffix to serve more than one function. In English, ***s*** can either be (1) a verb ending attached to third person singular forms (*runs*, *sings*), (2) a noun ending indicating the plural (*dogs*, *cats*) or (3) a noun ending indicating the possessive (*boy's*, *girls'*). By an orthographic convention now several hundred years old, the possessive is written with an apostrophe, but nowadays this is frequently omitted in familiar phrases (*a girls school*). (Usage (3) is relatively rare compared with (1) and (2): there are only nine uses of ***'s*** in this document.)

Since the normal order of suffixes is *d*, *i* and *a*, we can expect them to be removed from the right in the order *a*, *i* and *d*. Usually we want to remove all *a*- and *i*-suffixes, and some of the *d*-suffixes.

If the stemming process reduces two words to the same stem, they are said to be *conflated*.

3 Stemming errors, and the use of dictionaries

One way of thinking of the relation between terms and documents in an IR system is to see the documents as being about concepts, and the terms as words that describe the concepts. Then, of course, one word can cover many concepts, so *pound* can mean a unit of currency, a weight, an enclosure, or a beating. *Pound* is a homonym. And one concept can be described by many words, as with *money*, *capital*, *cash*, *currency*. These words are synonyms. There is a many-many mapping therefore between the set of terms and the set of concepts. Stemming is a process that transforms this mapping to advantage, on the whole reducing the number of synonyms, but occasionally creating new homonyms. It is worth remembering that what are called stemming errors are usually just the introduction of new homonyms into vocabularies that already contain very large numbers of homonyms.

Words which have no place in this term-concept mapping are those which describe no concepts. The particle words of grammar, *the*, *of*, *and* ..., known in IR as *stopwords*, fall into this category. Stopwords can be useful for retrieval but only in searching for phrases, *'to be or not to be'*, *'do as you would be done by'* etc. This suggests that stemming stopwords is not useful. More will be said on stopwords in section 7.

In the literature, a distinction is often made between under-stemming, which is the error of taking off too small a suffix, and over-stemming, which is the error of taking off too much. In French, for example, *croûtons* is the plural of *croûton*, 'a crust', so to remove ***ons*** would be over-stemming, while *croulons* is a verb form of *crouler*, 'to totter', so to remove ***s*** would be under-stemming. We would like to introduce a further distinction between mis-stemming and over-stemming. Mis-stemming is taking off what looks like an ending, but is really part of the stem. Over-stemming is taking off a true ending which results in the conflation of words of different meanings.

So for example ***ly*** can be removed from *cheaply*, but not from *reply*, because in *reply* ***ly*** is not a suffix. If it was removed, *reply* would conflate with *rep*, (the commonly used

short form of *representative*). Here we have a case of mis-stemming.

To illustrate over-stemming, look at these four words,

	verb	**adjective**
First pair:	prove	provable
Second pair:	probe	probable

Morphologically, the two pairs are exactly parallel (in the written, if not the spoken language). They also have a common etymology. All four words derive from the Latin *probare*, 'to prove or to test', and the idea of testing connects the meanings of the words. But the meanings are not parallel. *provable* means 'able to be proved'; *probable* does not mean 'able to be probed'. Most people would judge conflation of the first pair as correct, and of the second pair, incorrect. In other words, to remove ***able*** from *probable* is a case of over-stemming.

We can try to avoid mis-stemming and over-stemming by using a dictionary. The dictionary can tell us that *reply* does not derive from *rep*, and that the meanings of *probe* and *probable* are well separated in modern English. It is important to realise however that a dictionary does not give a complete solution here, but can be a tool to improve the conflation process.

In Krovetz's dictionary experiments (Krovetz 1995), he noted that in looking up a past participle like *suited*, one is led either to *suit* or to *suite* as plausible infinitive forms. *suite* can be rejected, however, because the dictionary tells us that although it is a word of English it is not a verb form. Cases like this (and Krovetz found about 60) had to be treated as exceptions. But the form *routed* could either derive from the the verb *rout* or the verb *route*:

> At Waterloo Napoleon's forces were routed
> The cars were routed off the motorway

Such cases in English are extremely rare, but they are commoner in more highly inflected languages. In French for example, *affiliez* can either be the verb *affiler*, to sharpen, with imperfect ending ***iez***, or the verb *affilier*, to affiliate, with present indicative ending ***ez***:

vous affiliez = vous affil-iez = you sharpened
vous affiliez = vous affili-ez = you affiliate

If the second is intended, removal of ***iez*** is mis-stemming.

With over-stemming we must rely upon the dictionary to separate meanings. There are different ways of doing this, but all involve some degree of reliance upon the lexicographers. Krovetz's methods are no doubt best, because the most objective: he uses several measures, but they are based on the idea of measuring the similarity in meaning of two words by the degree of overlap among the words used to define them, and this is at a good remove from a lexicographer's subjective judgement about semantic similarity.

There is an interesting difference between mis-stemming and over-stemming to do with language history. The morphology of a language changes less rapidly than the meanings of the words in it. When extended to include a few archaic endings, such as ***ick*** as an alternative to ***ic***, a stemmer for contemporary English can be applied to the English of 300 years ago. Mis-stemmings will be roughly the same, but the pattern of over-stemming will be different because of the changing meaning of words in the language. For example, *relativity* in the 19th century merely meant 'the condition of being relative to'. With that meaning, it is acceptable to conflate it with *relative*. But with the 20th century meaning brought to it by Einstein, stemming to *relativ* is over-stemming. Here we see the word with the suffix changing its meaning, but it can happen the other way round. *transpire* has come to mean 'happen', and its old meaning of 'exhalation' or 'breathing out' is now effectively lost. (That is the bitter reality, although dictionaries still try to persuade us otherwise). But *transpiration* still carries the earlier meaning. So what was formerly an acceptable stemming may be judged now as an over-stemming, not because the word being stemmed has changed its meaning, but because some cognate word has changed its meaning.

In these examples we are presenting words as if they had single meanings, but the true picture is more complicated. Krovetz uses a model of word meanings which is extremely helpful here. He makes a distinction between *homonyms* and *polysemes*. The meaning of homonyms are quite

unrelated. For example, *ground* in the sense of 'earth', and 'ground' as the past participle of 'grind' are homonyms. Etymologically homonyms have different stories, and they usually have separate entries in a dictionary. But each homonym form can have a range of polysemic forms, corresponding to different shades of meaning. So *ground* can mean the earth's surface, or the bottom of the sea, or soil, or any base, and so the basis of an argument, and so on. Over time new polysemes appear and old ones die. At any moment, the use of a word will be common in some polysemic forms and rare in others. If a suffix is attached to a word the new word will get a different set of polysemes. For example, *grounds* = *ground* + ***s*** acquires the sense of 'dregs' and 'estate lands', loses the sense of 'earth', and shares the sense of 'basis'. Consider the conflation of *mobility* with *mobile*. *mobile* has acquired two new polysemes not shared with *mobility*. One is the 'mobile art object', common in the nursery. This arrived in the 1960s, and is still in use. The other is the 'mobile phone' which is now very dominant, although it may decline in the future when it has been replaced by some new gadget with a different name. We might draw a graph of the degree of separation of the meanings of *mobility* and *mobile* against time, which would depend upon the number of polysemes and the intensity of their use. What seemed like a valid conflation of the two words in 1940 may seem to be invalid today.

In general therefore one can say that judgements about whether words are over-stemmed change with time as the meanings of words in the language change.

The use of a dictionary should reduce errors of mis-stemming and errors of over-stemming. And, for English at least, the mis-stemming errors should reduce well, even if there are problems with over-stemming errors. Of course, it depends on the quality of the dictionary. A dictionary will need to be very comprehensive, fully up-to-date, and with good word definitions to achieve the best results.

Historically, stemmers have often been thought of as either dictionary-based or algorithmic. The presentation of studies of stemming in the literature has perhaps helped to create this division. In the Lovins' stemmer the algorithmic description is central. In accounts of dictionary-based stemmers the emphasis tends to be on dictionary content

and structure, and IR effectiveness. Savoy's French stemmer (Savoy, 1993) is a good example of this. But the two approaches are not really distinct. An algorithmic stemmer can include long exception lists that are effectively mini-dictionaries, and a dictionary-based stemmer usually needs a process for removing at least *i*-suffixes to make the look-up in the dictionary possible. In fact in a language in which proper names are inflected (Latin, Finnish, Russian ...), a dictionary-based stemmer will need to remove *i*-suffixes independently of dictionary look-up, because the proper names will not of course be in the dictionary.

The stemmers available on the Snowball website are all purely algorithmic. They can be extended to include built-in exception lists, they could be used in combination with a full dictionary, but they are still presented here in their simplest possible form. Being purely algorithmic, they are, or ought to be, inferior to the performance of well-constructed dictionary-based stemmers. But they are still very useful, for the following reasons:

1) Algorithmic stemmers are (or can be made) very lean and very fast. The stemmers presented here generate code that will process about a million words in six seconds on a conventional 500MHz PC. Nowadays we can generate very large IR systems with quite modest resources, and tools that assist in this have value.

2) Despite the errors they can be seen to make, algorithmic stemmers still give good practical results. As Krovetz (1995) says in surprise of the algorithmic stemmer, 'Why does it do so well?' (page 89).

3) Dictionary-based stemmers require dictionary maintenance, to keep up with an ever-changing language, and this is actually quite a problem. It is not just that a dictionary created to assist stemming today will probably require major updating in a few years time, but that a dictionary in use for this purpose today may already be several years out of date.

We can hazard an answer to Krovetz's question, as to why algorithmic stemmers perform as well as they do, when they reveal so many cases of under-, over- and mis-stemming. Under-stemming is a fault, but by itself it will not degrade

the performance of an IR system. Because of under-stemming words may fail to conflate that ought to have conflated, but you are, in a sense, no worse off than you were before. Mis-stemming is more serious, but again mis-stemming does not really matter unless it leads to false conflations, and that frequently does not happen. For example, removing the ***ate*** ending in English, can result in useful conflations (*luxury*, *luxuriate*; *affection*, *affectionate*), but very often produces stems that are not English words (*enerv-ate*, *accommod-ate*, *deliber-ate* etc). In the literature, these are normally classed as stemming errors — overstemming — although in our nomenclature they are examples of mis-stemming. However these residual stems, *enerv*, *accommod*, *deliber* ... do not conflate with other word forms, and so behave in an IR system in the same way as if they still retained their ***ate*** ending. No false conflations arise, and so there is no over-stemming here.

To summarise, one can say that just as a word can be over-stemmed but not mis-stemmed (*relativity* -> *relative*), so it can be mis-stemmed but not over-stemmed (*enervate* -> *enerv*). And, of course, even over-stemming does not matter, if the over-stemmed word falsely conflates with other words that exist in the language, but are not encountered in the IR system which is being used.

Of the three types of error, over-stemming is the most important, and using a dictionary does not eliminate all over-stemmings, but does reduce their incidence.

4 Stemming as part of an indexing process

Stemming is part of a composite process of extracting words from text and turning them into index terms in an IR system. Because stemming is somewhat complex and specialised, it is usually studied in isolation. Even so, it cannot really be separated from other aspect of the indexing process:

1) What is a word? For indexing purposes, a word in a European language is a sequence of letters bounded by non-letters. But in English, an internal apostrophe does not split

a word, although it is not classed as a letter. The treatment of these word boundary characters affects the stemmer. For example, the Kraaij Pohlmann stemmer for Dutch (Kraaij, 1994, 1995) removes hyphen and treats apostrophe as part of the alphabet (so ***'s***, ***'tje*** and ***'je*** are three of their endings). The Dutch stemmer presented here assumes hyphen and apostrophe have already been removed from the word to be stemmed.

2) What is a letter? Clearly letters define words, but different languages use different letters, much confusion coming from the varied use of accented Roman letters.

English speakers, perhaps influenced by the ASCII character set, typically regard their alphabet of ***a*** to ***z*** as the norm, and other forms (for example, Danish ***å*** and ***ø***, or German ***ß***) as somewhat abnormal. But this is an insular point of view. In Italian, for example, the letters ***j***, ***k***, ***w***, ***x*** and ***y*** are not part of the alphabet, and are only seen in foreign words. We also tend to regard other alphabets as only used for isolated languages, and that is not strictly true. Cyrillic is used for a range of languages other than Russian, among which additional letters and accented forms abound.

In English, a broad definition of letter would be anything that could be accepted as a pronounceable element of a word. This would include accented Roman letters (*naïve*, *Fauré*), and certain ligature forms (*encyclopædia*). It would exclude letters of foreign alphabets, such as Greek and Cyrillic. The ***a*** to ***z*** alphabet is one of those where letters come in two styles, upper and lower case, which historically correspond (very roughly) to the shapes you get if you use a chisel or a pen. Across all languages, the exact relation of upper to lower case is not so easy to define. In Italian, for example, an accented lower case letter is sometimes represented in upper case by an the unaccented letter followed by an apostrophe. (I have seen this convention used in modern Italian news stories in machine readable form.)

In fact the Porter stemmer (which is for English) assumes the word being stemmed is unaccented and in lower case. More exactly, ***a***, ***e***, ***i***, ***o***, ***u***, and sometimes ***y***, are treated as vowels, and any other character gets treated as a consonant. Each stemmer presented here assumes some degree of

normalisation before it receives the word, which is roughly (*a*) put all letters into lower case, and (*b*) remove accents from letter-accent combinations that do not form part of the alphabet of the language. Each stemmer declares the letter-accent combinations for its language, and this can be used as a guide for the normalisation, but even so, we can see from the discussion above that (*a*) and (*b*) are not trivial operations, and need to be done with care.

(Incidentally, because the stemmers work on lower case words, turning letters to upper case is sometimes used internally for flagging purposes.)

3) Identifying stopwords. Invariant stopwords are more easily found before stemming is applied, but inflecting stopwords (for example, German *kein*, *keine*, *keinem*, *keinen* ...) may be easier to find after — because there are fewer forms. There is a case for building stopword identification into the stemming process. See section 7.

4) Conflating irregular forms. More will be said on this in section 6.

5 The use of stemmed words

The idea of how stemmed words might be employed in an IR system has evolved slightly over the years. The Lovins stemmer (Lovins 1968) was developed not for indexing document texts, but the subject terms attached to them. With queries stemmed in the same way, the user needed no special knowledge of the form of the subject terms. Rijsbergen (1979, Chapter 2) assumes document text analysis: stopwords are removed, the remaining words are stemmed, and the resulting set of stemmed word constitute the IR index (and this style of use is widespread today). More flexibility however is obtained by indexing *all* words in a text in an unstemmed form, and keeping a separate two-column relation which connects the words to their stemmed equivalents. The relation can be denoted by *R(s, w)*, which means that *s* is the stemmed form of word *w*. From the relation we can get, for any word *w*, its unique stemmed form, *stem(w)*, and for any stem *s*, the set of words, *words(s)*, that stem to *s*.

The user should not have to see the stemmed form of a word. If a list of stems is to be presented back for query expansion, in place of a stem, *s*, the user should be shown a single representative from the set *words(s)*, the one of highest frequency perhaps. The user should also be able to choose for the whole query, or at a lower level for each word in a query, whether or not it should be stemmed. In the absence of such choices, the system can make its own decisions. Perhaps single word queries would not undergo stemming; long queries would; stopwords would be removed except in phrases. In query expansion, the system would work with stemmed forms, ignoring stopwords.

Query expansion with stemming results in a much cleaner vocabulary list than without, and this is a main strength of using a stemming process.

A question arises: if the user never sees the stemmed form, does its appearance matter? The answer must be no, although the Porter stemmer tries to make the unstemmed forms guessable from the stemmed forms. For example, from *appropri* you can guess *appropriate*. At least, trying to achieve this effect acts as a useful control. Similarly with the other stemmers presented here, an attempt has been made to keep the appearance of the stemmed forms as familiar as possible.

6 Irregular grammatical forms

All languages contain irregularities, but to what extent should they be accommodated in a stemming algorithm? An English stemmer, for example, can convert regular plurals to singular form without difficulty (*boys*, *girls*, *hands* ...). Should it do the same with irregular plurals (*men*, *children*, *feet*, ...)? Here we have irregular cases with *i*-suffixes, but there are irregularities with *d*-suffixes, which Lovins calls 'spelling exceptions'. *absorb/absorption* and *conceive/conception* are examples of this. Etymologically, the explanation of the first is that the Latin root, *sorbere*, is an irregular verb, and of the second that the word *conceive* comes to us from the French rather than straight from the Latin. It is interesting that, even with no knowledge of the etymology, we do recognise the connection between the words.

Lovins tries to solve spelling exceptions by formulating general respelling rules (turn ***rpt*** into ***rb*** for example), but it might be easier to have simply a list of exceptional stems.

The Porter stemmer does not handle irregularities at all, but from the author's own experience, this has never been an area of complaint. Complaints in fact are always about false conflations, for example *new* and *news*.

Possibly Lovins was right in wanting to resolve *d*-suffix irregularities, and not being concerned about *i*-suffix irregularities. *i*-suffix irregularities in English go with short, old words, that are either in very common use (*man/men*, *woman/women*, *see/saw* ...) or are used only rarely (*ox/oxen*, *louse/lice*, *forsake/forsook* ...). The latter class can be ignored, and the former has its own problems which are not always solved by stemming. For example *man* is a verb, and *saw* can mean a cutting instrument, or, as a verb, can mean to use such an instrument. Conflation of these forms frequently leads to an error like mis-stemming therefore.

An algorithmic stemmer really needs holes where the irregular forms can be plugged in as necessary. This is more serviceable than attempting to embed special lists of these irregular forms into software.

7 Stopwords

We have suggested that stemming stopwords is not useful. There is a grammatical connection between *being* and *be*, but conflation of the two forms has little use in IR because they have no shared meaning that would entitle us to think of them as synonyms. *being* and *be* have a morphological connection as well, but that is not true of *am* and *was*, although they have a grammatical connection. Generally speaking, inflectional stopwords exhibit many irregularities, which means that stemming is not only not useful, but not possible, unless one builds into the stemmer tables of exceptions.

Switching from English to French, consider *être*, the equivalent form of *be*. It has about 40 different forms, including,

(and *suis* incidentally is a homonym, as part of the verb *suivre*.) Passing all forms through a rule-based stemmer creates something of a mess. An alternative approach is to recognise this group of words, and other groups, and take special action. The recognition could take place inside the stemmer, or be done before the stemmer is called. One special action would be to stem (perhaps one should say 'map') all the forms to a standard form, ETRE, to indicate that they are parts of the verb *être*. Deciding what to do with the term ETRE, and it would probably be to discard it, would be done outside the stemming process. Another special action would be to recognize a whole class of stopwords and simply discard them.

The strategy adopted will depend upon the underlying IR model, so what one needs is the flexibility to create modified forms of a standard stemmer. Usually we present Snowball stemmers in their unadorned form. Thereafter, the addition of stopword tables is quite easy.

8 Rare forms

Stemmers do not need to handle linguistic forms that turn up only very rarely, but in practice it is hard to design a stemmer with all rare forms eliminated without there appearing to be some gaps in the thinking. For this reason one should not worry too much about their occasional presence. For example, in contemporary Portuguese, use of the second person plural form of verbs has almost completely disappeared. Even so, endings for those forms are included in the Portuguese stemmer. They appear in all the grammar books, and will in any case be found in older texts. The habit of putting in rare forms to 'complete the picture' is well established, and usually passes unnoticed. An example is the list of English stopwords in van Rijsbergen (1979). This includes *yourselves*, by analogy with *himself*, *herself* etc., although *yourselves* is actually quite a rare word in English.

References

Farber DJ, Griswold RE and Polonsky IP (1964) SNOBOL, a string manipulation language. *Journal of the Association for Computing Machinery*, **11**: 21-30.

Griswold RE, Poage JF and Polonsky IP (1968) *The SNOBOL4 programming language*. Prentice-Hall, New Jersey.

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

Jesperson O (1921) *Language, its nature, origin and development*. George Allen & Unwin, London.

Kraaij W and Pohlmann R. (1994) Porter's stemming algorithm for Dutch. In Noordman LGM and de Vroomen WAM, eds. *Informatiewetenschap 1994: Wetenschappelijke bijdragen aan de derde STINFON Conferentie*, Tilburg, 1994. pp. 167-180.

Kraaij W and Pohlmann R (1995) Evaluation of a Dutch stemming algorithm. Rowley J, ed. *The New Review of Document and Text Management*, volume 1, Taylor Graham, London, 1995. pp. 25-43,

Krovetz B (1995) *Word sense disambiguation for large text databases*. PhD Thesis. Department of Computer Science, University of Massachusetts Amherst.

Lennon M, Pierce DS, Tarry BD and Willett P (1981) An evaluation of some conflation algorithms for information retrieval. *Journal of Information Science*, **3**: 177-183.

Lovins JB (1968) Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, **11**: 22-31.

Palmer FR (1965) *A linguistic study of the English verb*. Longmans, London.

Popovic M and Willett P (1990) Processing of documents

and queries in a Slovene language free text retrieval system. *Literary and Linguistic Computing*, **5**: 182-190.

Porter MF (1980) An algorithm for suffix stripping. *Program*, 14: 130-137.

Rijsbergen CJ (1979) *Information retrieval*. Second edition. Butterworths, London.

Savoy J (1993) Stemming of French words based on grammatical categories. *Journal of the American Society for Information Science*, **44**: 1-9.

Schinke R, Greengrass M, Robertson AM and Willett P (1996) A stemming algorithm for Latin text databases. *Journal of Documentation*, **52**: 172-187.