Snowball Stemming Manual

Dr. Martin Porter Richard Boulton

2025

Table of contents

Snowball: A language for stemming algorithms

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 (Schinke 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 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 (Palmer, 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 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 sharp-ened
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 approximately a sequence of letters bounded by non-letters, but some punctuation needs special consideration. For example, an internal apostrophe typically does not split a word.

2. What is a letter? Clearly letters define words, but different languages use different alphabets. Even only considering languages using the Latin alphabet there are different accented 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 (*na**ï**ve*, *Faur**é***), and certain ligature forms (*encyclop**æ**dia*). 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 the unaccented letter followed by an apostrophe (this seems to be because Italian keywords have keys for accented lower case letters but typing accented upper case letters is more complicated). This is even sometimes seen in published books.

3. The Snowball stemmers expect that the input words have been converted to lower case.

   (Incidentally, because the stemmers work on lower case words, some of the algorithms sometimes turn letters to upper case internally for flagging purposes.)

4. The algorithms for languages which use accented characters are written assuming that no additional accent stripping is done before or after stemming. If you strip accents you may cause conflation of unrelated words - for example, in French *châtiment* (punishment) and *chat* (cat) would produce the same stem if accents are stripped.

5. 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.

6. 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,

suis es sommes serez étaient fus furent sois été

(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.

Part I

Overviews

Chapter 1

Germanic language stemmers

Despite its inflexional complexities, German has quite a simple suffix structure, so that, if one ignores the almost intractable problems of compound words, separable verb prefixes, and prefixed and infixed ***ge***, an algorithmic stemmer can be made quite short. (Infixed ***zu*** can be removed algorithmically, but this minor feature is not shown here.) The umlaut in German is a regular feature of plural formation, so its removal is a natural feature of stemming, but this leads to certain false conflations (for example, *schön*, beautiful; *schon*, already).

By contrast, Dutch is inflexionally simple, but even so, this does not make for any great difference between the stemmers. A feature of Dutch that makes it markedly different from German is that the grammar of the written language has changed, and continues to change, relatively rapidly, and that it has assimilated a large and mixed foreign vocabulary with some of the accompanying foreign suffixes. Foreign words may, or may not, be transliterated into a Dutch style. Naturally these create problems in stemming. The stemmer here is intended for native words of contemporary Dutch.

In a Dutch noun, a vowel may double in the singular form (manen = moons, maan = moon). We attempt to solve this by undoubling the double vowel (Kraaij Pohlman by contrast attempt to double the single vowel). The endings ***je***, ***tje***, ***pje*** etc., although extremely common, are not stemmed. They are diminutives and can significantly alter word meaning.

1.1 A note on compound words

Famously, German allows for the formation of long compound words, written without spaces. For retrieval purposes, it is useful to be able to search on the parts of such words, as well as the on the complete words themselves. This is not just peculiar to German: Dutch, Danish, Norwegian, Swedish, Icelandic and Finnish have the same property. To split up compound words cannot be done without a dictionary, and the purely algorithmic stemmers presented here do not attempt it.

We would suggest, however, that the need for compound word splitting in these languages has been somewhat overstated. In the case of German:

1) There are many English compounds one would see no advantage in splitting,

blackberry	blackboard	rainbow	coastguard	....

Many German compounds are like this,

Bleistift (pencil)	=	Blei (lead) + Stift (stick)
Eisenbahn (railway)	=	Eisen (iron) + Bahn (road)
Unterseeboot (submarine)	=	under + sea + boat

2) Other compounds correspond to what in English one would want to do by phrase searching, so they are ready made for that purpose,

Gesundheitspflege	=	'health care'
Fachhochschule	=	'technical college'
Kunstmuseum	=	'museum of fine art'

3) In any case, longer compounds, especially involving personal names, are frequently hyphenated,

Heinrich-Heine-Universität

4) It is possible to construct participial adjectives of almost any length, but they are little used in contemporary German, and regarded now as poor style. As in English, very long words are not always to be taken too seriously. On the author's last visit to Germany, the longest word he had to struggle with was

Nasenspitzenwurzelentzündung

It means 'inflammation of the root of the tip of the nose', and comes from a cautionary tale for children.

Chapter 2

Romance language stemmers

The Romance languages have a wealth of different *i*-suffixes among the verb forms, and relatively few for the other parts of speech. In addition to this, many verbs exhibit irregularities. Many also have short stems, leading to dangers of over-stemming. The verb, therefore, tends to dominate initial thinking about stemming in these languages.

An algorithmic stemmer can usually reduce the multiple forms of a verb to at most two or three, and often just one. This is probably adequate for standard IR use, where the verb is used rather less than other parts of speech in short queries.

In French the verb endings ***ent*** and ***ons*** cannot be removed without unacceptable overstemming. The ***ons*** form is rarer, but ***ent*** forms are quite common, and will appear regularly throughout a stemmed vocabulary.

In Italian, the final vowel of nouns and adjectives indicates number and gender (*amico* is male friend, *amica* is female friend) and its removal is a necessary part of stemming, but the final vowel sometimes separates words of different meanings (*banco* is bench, *banca* is bank), which leads to some over-stemming.

The *d*-suffixes of all four languages follow a similar pattern. They can be tabulated as follows,

		French	Spanish	Portug.	Italian
noun	ANCE	***ance***	***anza***	***eza***	***anza***
adjective	IC	***ique***	***ico***	***ico***	***ico***
noun	ISM	***isme***	***ismo***	***ismo***	***ismo***
adjective	ABLE	***able***	***able***	***ável***	***abile***
adjective	IBLE	-	***ible***	***ível***	***ibile***
noun	IST	***iste***	***ista***	***ista***	***ista***
adjective	OUS	***eux***	***oso***	***oso***	***oso***
noun	MENT	***ment***	***amiento***	***amento***	***mente***
noun	ATOR	***ateur***	***ador***	***ador***	***attore***
noun	ATRESS	***atrice***	-	-	***atrice***
noun	ATION	***ation***	***ación***	***ação***	***azione***
noun	LOGY	***logie***	***logía***	***logía***	***logia***
noun	USION	***usion***	***ución***	***ución***	***uzione***
noun	ENCE	***ence***	***encia***	***ência***	***enza***
adjective	ENT	***ent***	***ente***	***ente***	***ente***
noun	ANCE	***ance***	***ancia***	***ância***	***anza***

noun	ANT	***ant***	***ante***	***ante***	***ante***
adverb	LY	(***e***)***ment***	(***a***)***mente***	(***a***)***mente***	(***a***)***mente***
noun	ITY	***ité***	***idad***	***idade***	***ità***
adjective	IVE	***if***	***ive***	***ivo***	***ivo***
verb	ATE	***at***	***at***	***at***	***at***

Equivalent English forms are shown in upper case. In English, ATE is a valid ending, but in the Romance languages it only exists in combinations. The endings can appear in a number of styles. In Italian, ***oso*** can also be ***osa***, ***osi*** or ***ose***, French ***ique*** becomes ***ic*** in combinations.

The important combining forms are summarised in the following picture:

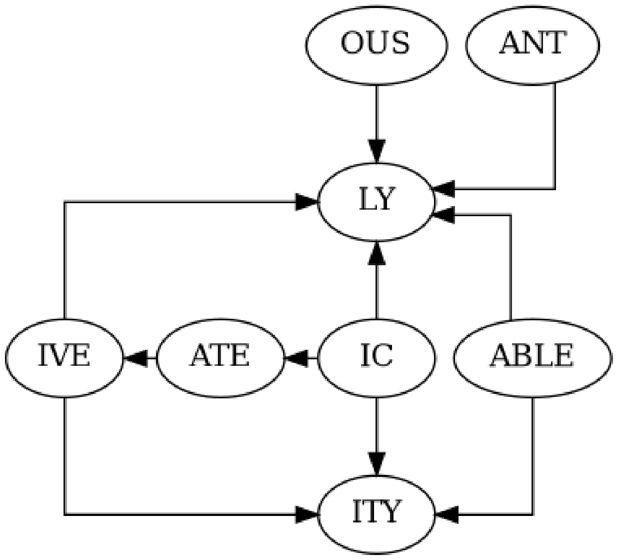

Figure 2.1: Graph showing important combining forms in English

In English, ABLE combines with LY to form ABLY. So in French, for example, ***able*** combines with (***e***)***ment*** to form ***ablement***. In some languages particular combinations are rare. In Italian, for example, ANT + LY, which would be the ending ***antemente***, is so rare that it does not figure in the stemming algorithm. According to the picture, we should encounter the forms ICATIVELY and ICATIVITY, and dictionaries instance a few English words with these endings (*communicatively* for example). But in practice three is the maximum number of derivational suffixes that one need consider in combination.

Chapter 3

Scandinavian language stemmers

The stemmers for the three Scandinavian languages are all very simple, and quite similar to each other. But between the languages there is a difference in which endings can be removed without difficulty, even though the endings are very similar. For example, in Norwegian the ending ***ede*** can be removed safely, but not in Danish.

The Scandinavian languages have a noun ending corresponding to the definite article (*the* in English). This ending cannot always be removed with certainty. In Swedish, for example, the ***en*** form is removed, and the ***et*** form in some cases, but not the ***t*** or ***n*** form,

husen		hus
valet		val
flickan	»	flickan
äpplet		äpplet

Chapter 4

Defining *R1* and *R2*

Most of the stemmers make use of at least one of the region definitions *R1* and *R2* They are defined as follows:

R1 is the region after the first non-vowel following a vowel, or is the null region at the end of the word if there is no such non-vowel.

R2 is the region after the first non-vowel following a vowel in *R1*, or is the null region at the end of the word if there is no such non-vowel.

The definition of *vowel* varies from language to language. In French, for example, ***é*** is a vowel, and in Italian ***i*** between two other vowels is not a vowel. The class of letters that constitute vowels is made clear in each stemmer.

Below, *R1* and *R2* are shown for a number of English words,

```
b   e   a   u   t   i   f   u   l
                     |<------------->|    R1
                             |<----->|    R2
```

Letter ***t*** is the first non-vowel following a vowel in *beautiful*, so *R1* is ***iful***. In ***iful***, the letter ***f*** is the first non-vowel following a vowel, so *R2* is ***ul***.

```
b   e   a   u   t   y
                     |<->|    R1
                       ->|<-  R2
```

In *beauty*, the last letter ***y*** is classed as a vowel. Again, letter ***t*** is the first non-vowel following a vowel, so *R1* is just the last letter, ***y***. *R1* contains no non-vowel, so *R2* is the null region at the end of the word.

```
b   e   a   u
             ->|<-  R1
             ->|<-  R2
```

In *beau*, *R1* and *R2* are both null.

Other examples:

```
a   n   i   m   a   d   v   e   r   s   i   o   n
     |<----------------------------------------->|    R1
             |<--------------------------------->|    R2
```

```
s   p   r   i   n   k   l   e   d
                  |<------------->|    R1
                               ->|<-  R2

e   u   c   h   a   r   i   s   t
          |<-------------------->|    R1
                      |<--------->|    R2
```

Chapter 5

Marking vowels as consonants

Some of the algorithms begin with a step which puts letters which are normally classed as vowels into upper case to indicate that they are are to be treated as consonants (the assumption being that the words are presented to the stemmers in lower case). Upper case therefore acts as a flag indicating a consonant.

For example, the English stemmer begins with the step

Set initial ***y***, or ***y*** after a vowel, to ***Y***,

giving rise to the following changes,

youth	»	*Youth*
boy	»	*boY*
boyish	»	*boYish*
fly	»	*fly*
flying	»	*flying*
syzygy	»	*syzygy*

This process works from left to right, and if a word contains ***Vyy***, where ***V*** is a vowel, the first ***y*** is put into upper case, but the second ***y*** is left alone, since it is preceded by upper case ***Y*** which is a consonant. A sequence ***Vyyyyy...*** would be changed to ***VYyYyY...***.

The combination ***yy*** never occurs in English, although it might appear in foreign words:

sayyid	»	*saYyid*

(A *sayyid*, my dictionary tells me, is a descendant of Mohammed's daughter Fatima.) But the left-to-right process is significant in other languages, for example French. In French the rule for marking vowels as consonants is,

Put into upper case ***u*** or ***i*** preceded and followed by a vowel, and ***y*** preceded or followed by a vowel. Put ***u*** after ***q*** into upper case.

which gives rise to,

ennuie	»	*ennuIe*

inquiétude	»	*inqUiétude*

In the first word, ***i*** is put into upper case since it has a vowel on both sides of it. In the second word, ***u*** after ***q*** is put into upper case, and again the following ***i*** is left alone, since it is preceded by upper case ***U*** which is a consonant.

Chapter 6

The apostrophe character

Representing apostrophe is problematical for various reasons,

1. There are two Unicode characters for apostrophe, U+0027 and U+2019. The former is also in both ASCII and ISO-8859-1 (Latin1) whereas the latter is not. Compare,

   Hamlet's father's ghost (U+0027)
   Hamlet’s father’s ghost (U+2019)

2. Although conceptually different from an apostrophe, a single closing quote is also represented by character U+2019.
3. Character U+0027 is used for apostrophe, single closing quote and single opening quote (U+2018).
4. A fourth character, U+201B, like U+2018 but with the tail ‘rising’ instead of ‘descending’, is also sometimes used as apostrophe (in the house style of certain publishers, for surnames like *M’Coy* and so on.)

In the English stemming algorithm, it is assumed that apostrophe is represented by U+0027. This makes it ASCII compatible. Clearly other codes for apostrophe can be mapped to this code prior to stemming.

In English orthography, apostrophe has one of three functions.

1. It indicates a contraction in what is now accepted as a single word: *o’clock*, *O’Reilly*, *M’Coy*. Except in proper names such forms are rare: the apostrophe in *Hallowe’en* is disappearing, and in *’bus* has disappeared.
2. It indicates a standard contraction with auxiliary or modal verbs: *you’re*, *isn’t*, *we’d*. There are about forty of these forms in contemporary English, and their use is increasing as they displace the full forms that were at one time used in formal documents. Although they can be reduced to word pairs, it is more convenient to treat them as single items (usually stopwords) in IR work. And then preserving the apostrophe is important, so that *he’ll*, *she’ll*, *we’ll* are not equated with *hell*, *shell*, *well* etc.
3. It is used to form the ‘English genitive’, *John’s book*, *the horses’ hooves* etc. This is a development of (1), where historically the apostrophe stood for an elided ***e***. (Similarly the printed form ***’d*** for ***ed*** was very common before the nineteenth century.) Although in decline (witness *pigs trotters*, *Girls School Trust*), its use continues in contemporary English, where it is fiercely promoted as correct grammar, despite (or it might be closer to the truth to say *because of*) its complete semantic redundancy.

For these reasons, the English stemmer treats apostrophe as if it were a letter, removing it from the beginning of a word, where it might have stood for an opening quote, from the end of the word, where it might have stood for a closing quote, or been an apostrophe following ***s***. The form ***’s*** is also treated as an ending.

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.

Part II

Algorithms

Chapter 7

Danish stemming algorithm

7.1 The stemming algorithm

The Danish alphabet includes the following additional letters,

æ å ø

The following letters are vowels:

a e i o u y æ å ø

A consonant is defined as a character from ASCII a-z which isn't a vowel (originally this was "A consonant is defined as a non-vowel" but since 2018-11-15 we've changed this definition to avoid the stemmer altering alphanumeric codes which end with a repeated digit).

R2 is not used: *R1* is defined in the same way as in the German stemmer. (See the note on *R1* and *R2*.)

Define a valid ***s***-ending as one of

a b c d f g h j k l m n o p r t v y z å

Do each of steps 1, 2, 3 and 4.

Step 1:

Search for the longest among the following suffixes in *R1*, and perform the action indicated.

(*a*) ***hed ethed ered e erede ende erende ene erne ere en heden eren er heder erer heds es endes erendes enes ernes eres ens hedens erens ers ets erets et eret***

delete

(*b*) ***s***

delete if preceded by a valid ***s***-ending

(Note that only the suffix needs to be in *R1*, the letter of the valid ***s***-ending is not required to be.)

Step 2:

Search for one of the following suffixes in *R1*, and if found delete the last letter.

gd dt gt kt

(For example, *friskt* » *frisk*)

Step 3:

If the word ends ***igst***, remove the final ***st***.

Search for the longest among the following suffixes in *R1*, and perform the action indicated.

(*a*) ***ig   lig   elig   els***

delete, and then repeat step 2

(*b*) ***løst***

replace with ***løs***

Step 4: undouble

If the word ends with double consonant in *R1*, remove one of the consonants.

(For example, *bestemmelse* » *bestemmels* (step 1) » *bestemm* (step *3a*) » *bestem* in this step.)

Chapter 8

Dutch stemming algorithm (Porter variant)

8.1 The stemming algorithm

Dutch includes the following accented forms

ä ë ï ö ü á é í ó ú è

First, remove all umlaut and acute accents listed above. A vowel is then one of,

a e i o u y è

Put initial ***y***, ***y*** after a vowel, and ***i*** between vowels into upper case. *R1* and *R2* (see the note on *R1* and *R2*) are then defined as in German.

Define a valid ***s***-ending as a non-vowel other than ***j***.

Define a valid ***en***-ending as a non-vowel, and not ***gem***.

Define undoubling the ending as removing the last letter if the word ends ***kk***, ***dd*** or ***tt***.

Do each of steps 1, 2 3 and 4.

Step 1:

Search for the longest among the following suffixes, and perform the action indicated

(*a*) ***heden***

replace with ***heid*** if in *R1*

(*b*) ***en ene***

delete if in *R1* and preceded by a valid ***en***-ending, and then undouble the ending

(*c*) ***s se***

delete if in *R1* and preceded by a valid ***s***-ending

Step 2:

Delete suffix ***e*** if in *R1* and preceded by a non-vowel, and then undouble the ending

Step 3a: ***heid***

delete ***heid*** if in *R2* and not preceded by ***c***, and treat a preceding ***en*** as in step 1(*b*)

Step 3b: *d*-suffixes

Search for the longest among the following suffixes, and perform the action indicated.

end   ing

delete if in *R2*

if preceded by ***ig***, delete if in *R2* and not preceded by ***e***, otherwise undouble the ending

ig

delete if in *R2* and not preceded by ***e***

lijk

delete if in *R2*, and then repeat step 2

baar

delete if in *R2*

bar

delete if in *R2* and if step 2 actually removed an ***e***

Step 4: undouble vowel

If the words ends *CVD*, where *C* is a non-vowel, *D* is a non-vowel other than ***I***, and *V* is double ***a***, ***e***, ***o*** or ***u***, remove one of the vowels from *V* (for example, *maan* » *man*, *brood* » *brod*).

Finally,

Turn ***I*** and ***Y*** back into lower case.

Chapter 9

Developing the English stemmer

(Revised slightly, December 2001)
(Further revised, September 2002)

Martin Porter made more than one attempt to improve the structure of the Porter algorithm by making it follow the pattern of ending removal of the Romance language stemmers. It is not hard to see why one should want to do this: step 1b of the Porter stemmer removes ***ed*** and ***ing***, which are *i*-suffixes attached to verbs. If these suffixes are removed, there should be no need to remove *d*-suffixes which are not verbal, although it will try to do so. This seems to be a deficiency in the Porter stemmer, not shared by the Romance stemmers. Again, the divisions between steps 2, 3 and 4 seem rather arbitrary, and are not found in the Romance stemmers.

Nevertheless, these attempts at improvement were abandoned. They seem to lead to a more complicated algorithm with no very obvious improvements. A reason for not taking note of the outcome of step 1b may be that English endings do not determine word categories quite as strongly as endings in the Romance languages. For example, *condition* and *position* in French have to be nouns, but in English they can be verbs as well as nouns,

We are all conditioned by advertising
They are positioning themselves differently today

A possible reason for having separate steps 2, 3 and 4 is that *d*-suffix combinations in English are quite complex, a point which has been made elsewhere.

But it is hardly surprising that after twenty years of use of the Porter stemmer, certain improvements did suggest themselves, and a new algorithm for English is therefore offered here. (It could be called the ‘Porter2’ stemmer to distinguish it from the Porter stemmer, from which it derives.) The changes are not so very extensive:

1. [In C Porter stemmer but not in paper] Extra rule in Step 2: ***logi*** -> ***log***
2. [In C Porter stemmer but not in paper] Step 2 rule: ***abli*** -> ***able*** replace by ***bli*** -> ***ble***
3. [In C Porter stemmer but not in paper] The algorithm leaves alone strings of length 2 (so ***as*** and ***is*** not longer lose ***s***.
4. Terminating ***y*** is changed to ***i*** rather less often
5. Suffix ***us*** does not lose its ***s***
6. A few additional suffixes are included for removal, including suffix ***ly***
7. A small list of exceptional forms is included
8. [December 2001] Steps 5a and 5b of the old Porter stemmer were combined into a single step. This means that undoubling final ***ll*** is not done with removal of final ***e***
9. [December 2001] In Step 3 ***ative*** is removed only when in region *R2*.

10. [September 2002] Exception added to prevent ***herring*** from stemming to ***her***.
11. [May 2005] ***commun*** added to exceptional forms
12. [July 2005] A small adjustment was made (including a new step 0) to handle apostrophe.
13. [January 2006] "Words" ***ied*** and ***ies*** now stem to ***ie*** rather than ***i***.
14. [January 2006] The implementation was fixed to follow the algorithm as documented here and now always treats an initial ***y*** as a consonant.
15. [November 2006] ***arsen*** added to exceptional forms
16. Snowball 3.0.0 Don't undouble if preceded by exactly ***a***, ***e*** or ***o***
17. Snowball 3.0.0 Exception added to prevent ***evening*** from stemming to ***even***.
18. Snowball 3.0.0 Removed exception for ***skis*** as the algorithm gives the same stem without it!
19. Snowball 3.0.0 Avoid conflating ***past*** with ***paste***/***pastes***/***pasted***/***pasting***.
20. Snowball 3.0.0 Avoid conflating ***universe***/***universes*** with ***universal***/***universally*** and ***university***/***universities***.
21. Snowball 3.0.0 Avoid conflating ***lateral***/***laterally*** with ***later***.
22. Snowball 3.0.0 Replace ***-ogist*** with ***-og*** to conflate ***geologist*** with ***geology***, etc.
23. Snowball 3.0.0 Handle ***-eed*** and ***-ing*** exceptions in respective rules.
24. Snowball 3.0.1 Restored exception for ***skis*** which **b** is needed.

9.1 Design Notes

Most comparatives (***-er***) are not handled. There is a rule to remove ***-er*** in *R2* (intended mostly for other forms with that ending rather than comparatives - e.g. *flatterer*, *observer*, *publisher*) which means longer comparatives are handled (e.g. *cleverer*, *yellower*) but most longer adjectives form the comparative with *more* e.g. *more attractive* rather than *attractiver*). Extending this rule to check *R1* as well would affect too many words where removal would be problematic - for example *charter*, *master*, *meter*, *number*, *mother*, *offer*, *proper*, *sober*, *solder*, *temper*, *wither*. The problem is worse in US English where many words which end ***-re*** in British English are instead spelled ***-er*** - e.g. *center*, *fiber*, *liter*, *luster*.

Superlatives are not handled at all due to too many cases where it would be problematic - for example *attest*, *behest*, *request*, *tempest*. There is not a rule to remove ***-est*** only in *R2* because it would help in very few cases but be harmful for e.g. *deforest*, *disinterest*, *interest*, *manifest*, *redigest*. These could be avoided by additional conditions on the removal, but the added complexity doesn't seem justifiable by the small number of words removing *-est* in *R2* would improve the stemming of.

Suffix *-ian* is not removed. A significant problem with removal is that sometimes we want to remove the whole three letters (e.g. *orwellian*, *politician*), sometimes *-an* (e.g. *comedian*, *historian*, *italian*) and sometimes just *-n* (e.g. *indian*, *bolivian*, *californian*).

9.2 Definition of the English stemmer

To begin with, here is the basic algorithm without reference to the exceptional forms. An exact comparison with the Porter algorithm needs to be done quite carefully if done at all. Here we indicate by * points of departure, and by + additional features. In the sample vocabulary, Porter and Porter2 stem slightly over 5% of words to different forms.

Define a *vowel* as one of

a e i o u y

Define a *double* as one of

bb dd ff gg mm nn pp rr tt

Define a *valid* **li**-*ending* as one of

c d e g h k m n r t

R1 is the region after the first non-vowel following a vowel, or the end of the word if there is no such non-vowel. (This definition may be modified for certain exceptional words — see below.)

R2 is the region after the first non-vowel following a vowel in *R1*, or the end of the word if there is no such non-vowel. (See note on *R1* and *R2*.)

Define a *short syllable* in a word as either (*a*) a vowel followed by a non-vowel other than ***w***, ***x*** or ***Y*** and preceded by a non-vowel, or * (*b*) a vowel at the beginning of the word followed by a non-vowel, or (*c*) ***past***.

So *rap*, *trap*, *entrap* end with a short syllable, and *ow*, *on*, *at*, *past* are classed as short syllables. But *uproot*, *bestow*, *disturb* do not end with a short syllable.

A word is called *short* if it ends in a short syllable, and if *R1* is null.

So *bed*, *shed* and *shred* are short words, *bead*, *embed*, *beds* are not short words.

An apostrophe (*'*) may be regarded as a letter. (See note on apostrophes in English.)

If the word has two letters or less, leave it as it is.

Otherwise, do each of the following operations,

Remove initial ***'***, if present. + Then,

Set initial ***y***, or ***y*** after a vowel, to ***Y***, and then establish the regions *R1* and *R2*. (See note on vowel marking.)

Step 0: +

Search for the longest among the suffixes,

'

's

's'

and remove if found.

Step 1a:

Search for the longest among the following suffixes, and perform the action indicated.

sses

replace by ***ss***

ied+ ***ies****

replace by ***i*** if preceded by more than one letter, otherwise by ***ie*** (so *ties* » *tie*, *cries* » *cri*)

s

delete if the preceding word part contains a vowel not immediately before the ***s*** (so *gas* and *this* retain the ***s***, *gaps* and *kiwis* lose it)

us+ ***ss***

do nothing

Step 1b:

Search for the longest among the following suffixes, and perform the action indicated.

eed ***eedly***+

replace by ***ee*** if in *R1*

ed   edly*+* ***ing   ingly****+*

for ***ing***, check if the word before the suffix is exactly one of the following exceptional cases:

* if it's a non-vowel followed by ***y***, replace ***y*** and ***ing*** with ***ie*** (so *dying* » *die*), then go to step 1c.

* if it's exactly one of *inn*, *out*, *cann*, *herr*, *earr* or *even* then go to step 1c.

delete if the preceding word part contains a vowel, and after the deletion:

if the word ends ***at***, ***bl*** or ***iz*** add ***e*** (so *luxuriat* » *luxuriate*), or

if the word ends with a *double* preceded by something other than exactly ***a***, ***e*** or ***o*** then remove the last letter (so *hopp* » *hop* but *add*, *egg* and *off* are not changed), or

if the word does not end with a *double* and is short, add ***e*** (so *hop* » *hope*)

Step 1c: *

replace suffix ***y*** or ***Y*** by ***i*** if preceded by a non-vowel which is not the first letter of the word (so *cry* » *cri*, *by* » *by*, *say* » *say*)

Step 2:

Search for the longest among the following suffixes, and, if found and in *R1*, perform the action indicated.

tional: replace by ***tion***

enci: replace by ***ence***

anci: replace by ***ance***

abli: replace by ***able***

entli: replace by ***ent***

izer   ization: replace by ***ize***

ational   ation   ator: replace by ***ate***

alism   aliti   alli: replace by ***al***

fulness: replace by ***ful***

ousli   ousness: replace by ***ous***

iveness   iviti: replace by ***ive***

biliti   bli*+*: replace by ***ble***

ogist*+*: replace by ***og***

ogi*+*: replace by ***og*** if preceded by ***l***

fulli*+*: replace by ***ful***

lessli*+*: replace by ***less***

li*+*: delete if preceded by a valid ***li***-ending

Step 3:

Search for the longest among the following suffixes, and, if found and in *R1*, perform the action indicated.

tional*+*: replace by ***tion***

ational+: replace by ***ate***

alize: replace by ***al***

icate ***iciti*** ***ical***: replace by ***ic***

ful ***ness***: delete

ative*: delete if in *R2*

Step 4:

Search for the longest among the following suffixes, and, if found and in *R2*, perform the action indicated.

al ***ance*** ***ence*** ***er*** ***ic*** ***able*** ***ible*** ***ant*** ***ement*** ***ment*** ***ent*** ***ism*** ***ate*** ***iti*** ***ous*** ***ive*** ***ize***

delete

ion

delete if preceded by ***s*** or ***t***

Step 5: *

Search for the following suffixes, and, if found, perform the action indicated.

e

delete if in *R2*, or in *R1* and not preceded by a short syllable

l

delete if in *R2* and preceded by ***l***

Finally, turn any remaining ***Y*** letters in the word back into lower case.

Chapter 10

Finnish stemming algorithm

10.1 Design Notes

Finnish is not an Indo-European language, but belongs to the Finno-Ugric group, which again belongs to the Uralic group. Distinctions between *a*-, *i*- and *d*-suffixes can be made in Finnish, but they are much less sharply separated than in an Indo-European language. The system of endings is extremely elaborate, but strictly defined, and applies equally to all nominals, that is, to nouns, adjectives and pronouns. Verb endings have a close similarity to nominal endings, which again makes Finnish very different from any Indo-European language.

More problematical than the endings themselves is the change that can be effected in a stem as a result of taking a particular ending. A stem typically has two forms, *strong* and *weak*, where one class of ending follows the strong form and the complementary class the weak. Normalising strong and weak forms after ending removal is not generally possible, although the common case where strong and weak forms only differ in the single or double form of a final consonant can be dealt with.

The letter ***å*** is also in the Finnish alphabet, but is only used in Scandanavian names. Treating it as a vowel in the stemmer does not improve retrieval results (instead it makes things very slightly worse as it will tend to cause the stemmer to conflate names with other words more).

10.2 The stemming algorithm

Finnish includes the following accented forms,

ä ö

The following letters are vowels:

a e i o u y ä ö

R1 and *R2* are then defined in the usual way (see the note on *R1* and *R2*).

Do each of steps 1, 2, 3, 4, 5 and 6.

Step 1: particles etc

Search for the longest among the following suffixes in *R1*, and perform the action indicated

(*a*) ***kin kaan kään ko kö han hän pa pä*** delete if preceded by ***n***, ***t*** or a vowel

(*b*) ***sti***

delete if in *R2*

Note that only the suffix needs to be in *R1*, the ***n***, ***t*** or vowel of 1(*a*) is not required to be. And similarly below.

Step 2: possessives

Search for the longest among the following suffixes in *R1*, and perform the action indicated

si

delete if not preceded by ***k***

ni

delete

if preceded by ***kse***, replace with ***ksi***

nsa nsä mme nne

delete

an

delete if preceded by one of ***ta ssa sta lla lta na***

än

delete if preceded by one of ***tä ssä stä llä ltä nä***

en

delete if preceded by one of ***lle ine***

The remaining steps require a few definitions.

Define a *v* (vowel) as one of ***a e i o u y ä ö***.
Define a *V* (restricted vowel) as one of ***a e i o u ä ö***.
So *V**i*** means a *V* followed by letter ***i***.
Define *LV* (long vowel) as one of ***aa ee ii oo uu ää öö***.
Define a *c* (consonant) as a character from ASCII a-z which isn't in *v* (originally this was "a character other than a *v* but since 2018-04-11 we've changed this definition to avoid the stemmer from altering sequences of digits).
So *cv* means a *c* followed by a *v*.

Step 3: cases

Search for the longest among the following suffixes in *R1*, and perform the action indicated

hXn preceded by *X*, where *X* is a *V* other than ***u*** (***a/han***, ***e/hen*** etc)

siin den tten preceded by *V**i***

seen preceded by *LV*

a ä preceded by *cv*

tta ttä preceded by ***e***

ta tä ssa ssä sta stä lla llä lta ltä lle na nä ksi ine

delete

n

delete, and if preceded by *LV* or ***ie***, delete the last vowel

So *aarteisiin* » *aartei*, the longest matching suffix being ***siin***, preceded as it is by *V**i***. But *adressiin* » *adressi*. The longest matching suffix is not ***siin***, because there is no preceding *V**i***, but ***n***, and then the last vowel of the preceding *LV* is removed.

Step 4: other endings

Search for the longest among the following suffixes in *R2*, and perform the action indicated

mpi mpa mpä mmi mma mmä

delete if not preceded by ***po***

impi impa impä immi imma immä eja ejä

delete

Step 5: plurals

If an ending was removed in step 3, delete a final ***i*** or ***j*** if in *R1*; otherwise, if an ending was not removed in step 3, delete a final ***t*** in *R1* if it follows a vowel, and, if a ***t*** is removed, delete a final ***mma*** or ***imma*** in *R2*, unless the ***mma*** is preceded by ***po***.

Step 6: tidying up

Do in turn steps (*a*), (*b*), (*c*), (*d*), restricting all tests to the region *R1*.

a) If *R1* ends *LV* delete the last letter
b) If *R1* ends *cX*, *c* a consonant and *X* one of ***a ä e i***, delete the last letter
c) If *R1* ends ***oj*** or ***uj*** delete the last letter
d) If *R1* ends ***jo*** delete the last letter

Do step (*e*), which is not restricted to *R1*.

e) If the word ends with a double consonant followed by zero or more vowels, remove the last consonant (so *eläkk* » *eläk*, *aatonaatto* » *aatonaato*)

Chapter 11

French stemming algorithm

11.1 Design Notes

In French the verb endings ***-ent*** and ***-ons*** cannot be removed without unacceptable overstemming (of *accident* and *garçons* for example). The ***-ons*** form is rarer, but ***-ent*** forms are quite common, and will appear regularly throughout a stemmed vocabulary.

The rule to replace ***-oux*** with ***-ou*** will produce linguistically incorrect stems by removing ***x*** from ***époux*** and ***jaloux***, but this is harmless as no unwanted conflation results.

The rule to replace ***-aux*** with ***-al*** stems ***travaux*** to ***traval*** and ***vitraux*** to ***vitral*** wheras ***-ail*** would be lingusitically correct; similarly ***esquimaux***, ***noyaux*** and ***tuyaux*** where ***-au*** would be more correct. None of these seem to result in unwanted conflation though.

The suffix ***-eux*** is removed if in *R2*. For a few short words (***cheveux***, ***jeux***, ***lieux*** and ***neveux***) removing ***-x*** would conflate a plural form with its singular, but the complexity of the rule needed to avoid adversely affecting other short words ending ***-eux*** does not seem justified for just four cases of understemming.

11.2 The stemming algorithm

Letters in French include the following accented forms,

â à ç ë é ê è ï î ô û ù

The following letters are vowels:

a e i o u y â à ë é ê è ï î ô û ù

The first step removes elisions. If the word starts with one of ***c d j l m n s t*** or ***qu***, followed by an apostrophe (***'***) which is not at the end of the word, then remove from the prefix of the word up to and including this apostrophe.

Assume the word is in lower case. Then, taking the letters in turn from the beginning to end of the word, put ***u*** or ***i*** into upper case when it is both preceded and followed by a vowel; put ***y*** into upper case when it is either preceded or followed by a vowel; and put ***u*** into upper case when it follows ***q***. For example,

jouer	»	joUer

ennuie	»	ennuIe
yeux	»	Yeux
quand	»	qUand
croyiez	»	croYiez

In the last example, ***y*** becomes ***Y*** because it is between two vowels, but ***i*** does not become ***I*** because it is between ***Y*** and ***e***, and ***Y*** is not defined as a vowel above.

(The upper case forms are not then classed as vowels — see note on vowel marking.)

Replace ***ë*** and ***ï*** with ***He*** and ***Hi***. The ***H*** marks the vowel as having originally had a diaeresis, while the vowel itself, lacking an accent, is able to match suffixes beginning in ***e*** or ***i***.

RV is defined as the region to the right of the first of these which is true (the examples show *RV* underlined):

1. The word starts with two vowels followed by another letter
   - e.g. *aimer*
2. The word starts ***par***
   - e.g. *parie*
3. The word starts ***col***
   - e.g. *colis*
4. The word starts ***tap***
   - e.g. *tapis*
5. The word starts ***ni*** followed by any vowel
   - e.g. *niaises*
6. The first vowel not at the beginning of the word
   - e.g. *adorer*, *voler*
7. The end of the word - e.g. *arcs* (RV is an empty region at the end of the word)

R1 is the region after the first non-vowel following a vowel, or the end of the word if there is no such non-vowel.

R2 is the region after the first non-vowel following a vowel in *R1*, or the end of the word if there is no such non-vowel. (See note on *R1* and *R2*.)

For example:

```
f a m e u s e m e n t
     |......R1.......|
           |...R2....|
```

Note that *R1* can contain *RV* (*adorer*), and *RV* can contain *R1* (*voler*).

Below, 'delete if in *R2*' means that a found suffix should be removed if it lies entirely in *R2*, but not if it overlaps *R2* and the rest of the word. 'delete if in *R1* and preceded by *X*' means that *X* itself does not have to come in *R1*, while 'delete if preceded by *X* in *R1*' means that *X*, like the suffix, must be entirely in *R1*.

Start with step 1

Step 1: Standard suffix removal

Search for the longest among the following suffixes, and perform the action indicated.

ance iqUe isme able iste eux ances iqUes ismes ables istes

delete if in *R2*

atrice ateur ation atrices ateurs ations

delete if in *R2*

if preceded by ***ic***, delete if in *R2*, else replace by ***iqU***

logie logies

replace with ***log*** if in *R2*

usion ution usions utions

replace with ***u*** if in *R2*

ence ences

replace with ***ent*** if in *R2*

ement ements

delete if in *RV*

if preceded by ***iv***, delete if in *R2* (and if further preceded by ***at***, delete if in *R2*), otherwise,

if preceded by ***eus***, delete if in *R2*, else replace by ***eux*** if in *R1*, otherwise,

if preceded by ***abl*** or ***iqU***, delete if in *R2*, otherwise,

if preceded by ***ièr*** or ***Ièr***, replace by ***i*** if in *RV*

ité ités

delete if in *R2*

if preceded by ***abil***, delete if in *R2*, else replace by ***abl***, otherwise,

if preceded by ***ic***, delete if in *R2*, else replace by ***iqU***, otherwise,

if preceded by ***iv***, delete if in *R2*

if ive ifs ives

delete if in *R2*

if preceded by ***at***, delete if in *R2* (and if further preceded by ***ic***, delete if in *R2*, else replace by ***iqU***)

eaux

replace with ***eau***

aux

replace with ***al*** if in *R1*

oux

replace with ***ou*** if preceded by one of ***b h j l n p***

euse euses

delete if in *R2*, else replace by ***eux*** if in *R1*

issement issements

delete if in *R1* and preceded by a non-vowel

amment

replace with ***ant*** if in *RV*

emment

replace with ***ent*** if in *RV*

ment ments

delete if preceded by a vowel in *RV*

Do step *2a* if either no ending was removed by step 1, or if one of endings ***amment***, ***emment***, ***ment***, ***ments*** was found.

Step *2a*: Verb suffixes beginning ***i***

Search for the longest among the following suffixes in RV and if found, delete if the preceding character is also in RV and is neither a vowel nor ***H***.

îmes ît îtes i ie ies ir ira irai iraIent irais irait iras irent irez iriez irions irons iront is issaIent issais issait issant issante issantes issants isse issent isses issez issiez issions issons it

Do step *2b* if step *2a* was done, but failed to remove a suffix.

Step *2b*: Other verb suffixes

Search for the longest among the following suffixes in RV, and perform the action indicated.

ions

delete if in *R2*

é ée ées és èrent er era erai eraIent erais erait eras erez eriez erions erons eront ez iez

delete

âmes ât âtes a ai aIent ait ant ante antes ants as asse assent asses assiez assions

delete

if preceded by ***e*** which is also in RV, delete the ***e*** as well

ais aise aises

delete unless preceded by one of: ***al*** preceded by exactly one character, ***auv***, ***épl***

If the last step to be obeyed — either step 1, *2a* or *2b* — altered the word, do step 3

Step 3

Replace final ***Y*** with ***i*** or final ***ç*** with ***c***

Alternatively, if the last step to be obeyed did not alter the word, do step 4

Step 4: Residual suffix

If the word ends ***s***, not preceded by ***a***, ***i*** (unless itself preceded by ***H***), ***o***, ***u***, ***è*** or ***s***, delete it.

In the rest of step 4, all tests are confined to the *RV* region.

Search for the longest among the following suffixes, and perform the action indicated.

ion

delete if in *R2* and preceded by ***s*** or ***t***

ier ière Ier Ière

replace with ***i***

e

delete

(So note that ***ion*** is removed only when it is in *R2* — as well as being in *RV* — and preceded by ***s*** or ***t*** which must be in *RV*.)

Always do steps 5 and 6.

Step 5: Undouble

If the word ends ***enn***, ***onn***, ***ett***, ***ell*** or ***eill***, delete the last letter

Step 6: Un-accent

If the words ends ***é*** or ***è*** followed by at least one non-vowel, remove the accent from the ***e***.

And finally:

Turn any remaining ***I***, ***U*** and ***Y*** letters in the word back into lower case.

Turn ***He*** and ***Hi*** back into ***ë*** and ***ï***, and remove any remaining ***H***.

11.3 History of functional changes to the algorithm

- September 2002: New rule for *-ièr*
- Snowball 2.0.0: Suffixes that begin with a diaereses are now removed (done by replacing ***ë*** and ***ï*** with ***He*** and ***Hi***, during stemming then undoing afterwards).
- Snowball 3.0.0: Added a new first step which removes elisions.
- Snowball 3.0.0: Added rule to replace ***-oux*** with ***-ou***.
- Snowball 3.0.0: Added RV exceptions for ***ni*** followed by a vowel.
- Snowball 3.0.0: Restrict removal of ***-ais***; remove ***-aise*** and ***-aises***.

Chapter 12

German stemming algorithm

12.1 Design Notes

Despite its inflexional complexities, German has quite a simple suffix structure, so that, if one ignores the almost intractable problems of compound words, separable verb prefixes, and prefixed and infixed ***ge***, an algorithmic stemmer can be made quite short. (Infixed ***zu*** can be removed algorithmically, but this minor feature is not shown here.) The umlaut in German is a regular feature of plural formation, so its removal is a natural feature of stemming, but this leads to certain false conflations (for example, *schön*, beautiful; *schon*, already).

There are a few short suffixes (for example, *-t*) where removal would cause problems so the stemmer leaves these alone - for example *holen*, *hole*, *hol* and *holest* are stemmed to *hol*; ideally *holt* would be too, but a rule to remove *-t* would adversely affect too many other words.

For similar reasons, some suffixes are only removed when in *R2*: for example *-end*.

Suffixes *-et*, *-s* and *-st* are removed in some cases. The stemmer checks the end of the stem which would be left to determine when to remove these suffixes, erring on the side of non-removal when it might be problematic.

As with the other stemmers, words are assumed to be lower cased before stemming. This potentially loses information since nouns are always capitalised in German, which results in some possibly avoidable conflations (e.g. *Planet* means "planet" but *planet* is a form of the verb "to plan"). However words are also capitalised for other reasons (e.g. at the start of a sentence or in a title) so capitalisation is not a completely reliable indicator.

In German, ***ä***, ***ö***, ***ü*** and ***ß*** are sometimes transliterated as ***ae***, ***oe***, ***ue*** and ***ss*** respectively. There is now a capital version of ***ss***, but it's a fairly recent invention (added to Unicode in 2008) and prior to this in capitalised text such as on street signs ***SS*** was used instead. Swiss German always uses ***ss*** instead of ***ß***, and also uses the transliterated forms for capital letters with umlauts. The German spelling reform of 1996 also reduced use of ***ß***, replacing it with ***ss*** in some words. Finally (but much less relevant nowadays) these transliterations are used when writing in character sets which lack these characters, or with a keyboard lacking an easy way to type them. The stemmer will conflate words spelled using these characters with those spelled using the transliterated versions.

Compound words

Famously, German allows for the formation of long compound words, written without spaces. For retrieval purposes, it is useful to be able to search on the parts of such words, as well as the on the complete words themselves. This is not just peculiar to German: Dutch, Danish, Norwegian, Swedish, Icelandic and Finnish have the same property. To split up compound words cannot be done without a dictionary, and the purely algorithmic stemmers presented here do not attempt it.

We would suggest, however, that the need for compound word splitting in these languages has been somewhat overstated. In the case of German:

1. There are many English compounds one would see no advantage in splitting,

blackberry	blackboard	rainbow	coastguard	....

   Many German compounds are like this,

Bleistift (pencil)	=	Blei (lead) + Stift (stick)
Eisenbahn (railway)	=	Eisen (iron) + Bahn (road)
Unterseeboot (submarine)	=	under + sea + boat

2. Other compounds correspond to what in English one would want to do by phrase searching, so they are ready made for that purpose,

Gesundheitspflege	=	‘health care’
Fachhochschule	=	‘technical college’
Kunstmuseum	=	‘museum of fine art’

3. In any case, longer compounds, especially involving personal names, are frequently hyphenated,

   Heinrich-Heine-Universität

4. It is possible to construct participial adjectives of almost any length, but they are little used in contemporary German, and regarded now as poor style.

12.2 History of functional changes to the algorithm

- Snowball 2.0.0 (2009-12-11): Extra rule for -nisse ending added
- Snowball 3.0.0: Handle ASCII transliterations of umlauts (merging the “german2” variant into the standard algorithm).
- Snowball 3.0.0: Special case for ***-system*** added.
- Snowball 3.0.0: Replace ***-ln*** and ***-lns*** with ***l***.
- Snowball 3.0.0: Remove ***-erin*** and ***-erinnen***.
- Snowball 3.0.0: Remove ***-et*** when safe to do so.

12.3 The stemming algorithm

German includes the following accented forms,

ä ö ü

and a special letter, ***ß***, equivalent to double ***s***.

The following letters are vowels:

a e i o u y ä ö ü

First put ***u*** and ***y*** between vowels into upper case, and then do the following mappings,

(*a*) replace ***ß*** with ***ss***,
(*b*) replace ***ae*** with ***ä***,
(*c*) replace ***oe*** with ***ö***,
(*d*) replace ***ue*** with ***ü*** unless preceded by ***q***.

(The rules here for ***ae***, ***oe*** and ***ue*** were added in Snowball 3.0.0, but were previously present as a variant of the algorithm termed "german2"). The condition on the replacement of ***ue*** prevents the unwanted changing of *quelle*. Also note that *feuer* is not modified because the first part of the rule changes it to *feUer*, so ***ue*** is not found.)

R1 and *R2* are first set up in the standard way (see the note on *R1* and *R2*), but then *R1* is adjusted so that the region before it contains at least 3 letters.

Define a valid ***s***-ending as one of ***b***, ***d***, ***f***, ***g***, ***h***, ***k***, ***l***, ***m***, ***n***, ***r*** or ***t***.

Define a valid ***st***-ending as the same list, excluding letter ***r***.

Define a valid ***et***-ending as one of ***d***, ***f***, ***g***, ***k***, ***l***, ***m***, ***n***, ***r***, ***s***, ***t***, ***U***, ***z*** or ***ä***.

Do each of steps 1, 2 and 3.

Step 1:

Search for the longest among the following suffixes,

(*a*) ***em*** (not preceded by ***syst*** [condition added in Snowball 3.0.0])
(*b*) ***ern*** ***er***
(*c*) ***e*** ***en*** ***es***
(*d*) ***s*** (preceded by a valid ***s***-ending)
(*e*) ***erin*** ***erinnen*** [added in Snowball 3.0.0]
(*f*) ***ln*** ***lns*** [added in Snowball 3.0.0]

and if in *R1* then delete (for (a) to (e)) or replace with *l* (for (f)). (Note that only the suffix needs to be in *R1*, the letter of the valid ***s***-ending is not required to be.)

If an ending of group (*c*) is deleted, and the ending is preceded by ***niss***, delete the final ***s***.

(For example, *äckern* » *äck*, *ackers* » *acker*, *armes* » *arm*, *bedürfnissen* » *bedürfnis*)

Step 2:

Search for the longest among the following suffixes,

(*a*) ***en*** ***er*** ***est***
(*b*) ***st*** (preceded by a valid ***st***-ending, itself preceded by at least 3 (*c*) ***et*** (preceded by a valid ***et***-ending, itself not preceded any of ***geordn***, ***intern***, ***plan***, ***tick*** or ***tr***).

and delete the suffix if in *R1*.

(For example, *derbsten* » *derbst* by step 1, and *derbst* » *derb* by step 2, since ***b*** is a valid ***st***-ending, and is preceded by just 3 letters)

Step 3: *d*-suffixes

Search for the longest among the following suffixes, and perform the action indicated.

end ***ung***

delete if in *R2*

if preceded by ***ig***, delete if in *R2* and not preceded by ***e***

ig ***ik*** ***isch***

delete if in *R2* and not preceded by ***e***

lich   ***heit***

delete if in *R2*

if preceded by ***er*** or ***en***, delete if in *R1*

keit

delete if in *R2*

if preceded by ***lich*** or ***ig***, delete if in *R2*

Finally,

turn ***U*** and ***Y*** back into lower case, and remove the umlaut accent from ***a***, ***o*** and ***u***.

Chapter 13

Italian stemming algorithm

13.1 The stemming algorithm

Italian can include the following accented forms:

á é í ó ú à è ì ò ù

First, replace all acute accents by grave accents. And, as in French, put ***u*** after ***q***, and ***u***, ***i*** between vowels into upper case. (See note on vowel marking.)

The vowels are then

a e i o u à è ì ò ù

R2 (see the note on *R1* and *R2*) and *RV* have the same definition as in the Spanish stemmer.

R2 is defined in the usual way — see the note on *R1* and *R2*.

RV is defined as follows (this is the same as the Spanish stemmer definition, except for the initial exceptional case):

If the word begins *divan* then *RV* starts after this prefix. If the second letter is a consonant, *RV* is the region after the next following vowel, or if the first two letters are vowels, *RV* is the region after the next consonant, and otherwise (consonant-vowel case) *RV* is the region after the third letter. But *RV* is the end of the word if these positions cannot be found.

Always do steps 0 and 1.

Step 0: Attached pronoun

Search for the longest among the following suffixes

ci gli la le li lo mi ne si ti vi sene gliela gliele glieli glielo gliene mela mele meli melo mene tela tele teli telo tene cela cele celi celo cene vela vele veli velo vene

following one of

(*a*) ***ando endo***
(*b*) ***ar er ir***

in *RV*. In case of (*a*) the suffix is deleted, in case
(*b*) it is replace by ***e***
(*guardandogli* » *guardando*, *accomodarci* » *accomodare*)

Step 1: Standard suffix removal

Search for the longest among the following suffixes, and perform the action indicated.

anza anze ico ici ica ice iche ichi ismo ismi abile abili ibile ibili ista iste isti istà istè istì oso osi osa ose mente atrice atrici ante anti

delete if in *R2*

azione azioni atore atori

delete if in *R2*; if preceded by ***ic*** which is also in *R2*, delete that too

logia logie

replace with ***log*** if in *R2*

uzione uzioni usione usioni

replace with ***u*** if in *R2*

enza enze

replace with ***ente*** if in *R2*

amento amenti imento imenti

delete if in *RV*

amente

delete if in *R1*

if preceded by ***iv***, delete if in *R2* (and if further preceded by ***at***, delete if in *R2*), otherwise,

if preceded by ***os***, ***ic*** or ***abil***, delete if in *R2*

ità

delete if in *R2*

if preceded by ***abil***, ***ic*** or ***iv***, delete if in *R2*

ivo ivi iva ive

delete if in *R2*

if preceded by ***at***, delete if in *R2* (and if further preceded by ***ic***, delete if in *R2*)

Do step 2 if no ending was removed by step 1.

Step 2: Verb suffixes

Search for the longest among the following suffixes in *RV*, and if found, delete.

ammo ando ano are arono asse assero assi assimo ata ate ati ato ava avamo avano avate avi avo emmo enda ende endi endo erà erai eranno ere erebbe erebbero erei eremmo eremo ereste eresti erete erò erono essero ete eva evamo evano evate evi evo Yamo iamo immo irà irai iranno ire irebbe irebbero irei iremmo iremo ireste iresti irete irò irono isca iscano isce isci isco iscono issero ita ite iti ito iva ivamo ivano ivate ivi ivo ono uta ute uti uto ar ir

Always do steps 3a and 3b.

Step 3a

Delete a final ***a***, ***e***, ***i***, ***o***, ***à***, ***è***, ***ì*** or ***ò*** if it is in *RV*, and a preceding ***i*** if it is in *RV* (*crocchi* » *crocch*, *crocchio* » *crocch*)

Step 3b

Replace final ***ch*** (or ***gh***) with ***c*** (or ***g***) if in *RV* (*crocch* » *crocc*)

Finally,

turn ***I*** and ***U*** back into lower case

13.2 History of functional changes to the algorithm

- 2005-06-15: Remove suffixes -*ante* and -*anti*.
- Snowball 3.0.0: Add exception to *RV* definition to avoid overstemming *divano*.

Chapter 14

Norwegian stemming algorithm

14.1 The stemming algorithm

The Norwegian alphabet includes the following additional letters,

æ å ø

The following letters are vowels:

a e ê i o ò ó ô u y æ å ø

R2 is not used: *R1* is defined in the same way as in the German stemmer. (See the note on *R1* and *R2*.)

Define a valid ***s***-ending as one of

b c d f g h j l m n o p t v y z,
or ***r*** not preceded by ***e***,
or ***k*** not preceded by a vowel.

Do each of steps 1, 2 and 3.

Step 1:

Search for the longest among the following suffixes in *R1*, and perform the action indicated.

(*a*) ***a e ede ande ende ane ene hetene en heten ar er heter as es edes endes enes hetenes ens hetens ets et het ast***

delete

(*b*) ***ers***

find the longest suffix preceding ***ers***, and perform the action indicated.

(*i*> ***amm ast ind kap kk lt nk omm pp v øst***

do nothing

(*ii*> ***giv hav skap***

delete ***ers*** suffix

(*c*) ***s***

delete if preceded by a valid ***s***-ending (*d*) ***erte   ert***

replace with ***er***

(Note that only the suffix needs to be in *R1*, the letter of the valid ***s***-ending is not required to be.)

Step 2:

If the word ends ***dt*** or ***vt*** in *R1*, delete the ***t***.

(For example, *meldt* » *meld*, *operativt* » *operativ*)

Step 3:

Search for the longest among the following suffixes in *R1*, and if found, delete.

leg   eleg   ig   eig   lig   elig   els   lov   elov   slov   hetslov

14.2 Design Notes

This algorithm aims to stem both Bokmål and Nynorsk, which are the two legally-recognised forms of written Norwegian.

Some other accented vowels are used in a small number of Norwegian words but these are deliberately not included in the list of vowels for this algorithm. This is not due to the small number of affected words but because including them doesn't actually improve the results of stemming. In most cases it would make no difference, but for ***é*** the reasoning is more subtle. Including it would make one difference - it would conflate forms of ***léta*** (to paint) but ***lét*** is both the imperative of ***léta*** and the past tense of ***la*** (to let/allow) so overall this conflation doesn't seem an improvement.

14.3 History of functional changes to the algorithm

- Snowball 2.0.0: ***s***-ending definition adjusted to only include ***k*** when preceded by a non-vowel.
- Snowball 3.0.0: Improve handling of words ending ***ers***.
- Snowball 3.0.0: Include ***ê***, ***ò***, ***ó*** and ***ô*** in the list of vowels.

Chapter 15

Portuguese stemming algorithm

15.1 Design Notes

This stemming algorithm aims to work with both European Portuguese and Brazilian Portuguese. It does not specifically attempt to normalise differences between the two, though removing suffixes will naturally tend to do this to some extent.

15.2 History of functional changes to the algorithm

- 2005-06-15: Added rules to remove *-ante*, *-antes* and *-ância*

15.3 The stemming algorithm

Letters in Portuguese include the following accented forms,

á é í ó ú â ê ô ç ã õ ü

The following letters are vowels:

a e i o u á é í ó ú â ê ô

And the two nasalised vowel forms,

ã õ

should be treated as a vowel followed by a consonant.

ã and ***õ*** are therefore replaced by ***a***~ and ***o***~ in the word, where ~ is a separate character to be treated as a consonant. And then —

R2 (see the note on *R1* and *R2*) and *RV* have the same definition as in the Spanish stemmer.

Always do step 1.

Step 1: Standard suffix removal

Search for the longest among the following suffixes, and perform the action indicated.

eza ezas ico ica icos icas ismo ismos ável ível ista istas oso osa osos osas amento amentos imento imentos adora ador aça~o adoras adores aço~es ante antes ância

delete if in *R2*

logia logias

replace with ***log*** if in *R2*

ução uções

replace with ***u*** if in *R2*

ência ências

replace with ***ente*** if in *R2*

amente

delete if in *R1*

if preceded by ***iv***, delete if in *R2* (and if further preceded by ***at***, delete if in *R2*), otherwise,

if preceded by ***os***, ***ic*** or ***ad***, delete if in *R2*

mente

delete if in *R2*

if preceded by ***ante***, ***avel*** or ***ível***, delete if in *R2*

idade idades

delete if in *R2*

if preceded by ***abil***, ***ic*** or ***iv***, delete if in *R2*

iva ivo ivas ivos

delete if in *R2*

if preceded by ***at***, delete if in *R2*

ira iras

replace with ***ir*** if in *RV* and preceded by ***e***

Do step 2 if no ending was removed by step 1.

Step 2: Verb suffixes

Search for the longest among the following suffixes in *RV*, and if found, delete.

ada ida ia aria eria iria ará ara erá era irá ava asse esse isse aste este iste ei arei erei irei am iam ariam eriam iriam aram eram iram avam em arem erem irem assem essem issem ado ido ando endo indo ara~o era~o ira~o ar er ir as adas idas ias arias erias irias arás aras erás eras irás avas es ardes erdes irdes ares eres ires asses esses isses astes estes istes is ais eis íeis aríeis eríeis iríeis áreis areis éreis ereis íreis ireis ásseis ésseis ísseis áveis ados idos ámos amos íamos aríamos eríamos iríamos áramos éramos íramos ávamos emos aremos eremos iremos ássemos êssemos íssemos imos armos ermos irmos eu iu ou ira iras

If the last step to be obeyed — either step 1 or 2 — altered the word, do step 3

Step 3

Delete suffix ***i*** if in *RV* and preceded by ***c***

Alternatively, if neither steps 1 nor 2 altered the word, do step 4

Step 4: Residual suffix

```
    If the word ends with one of the suffixes
        os   a   i   o   á   í   ó
            in RV, delete it
Always do step 5
Step 5:
    If the word ends with one of
        e   é   ê
            in RV, delete it, and if preceded by gu (or ci) with the u (or i) in RV, delete the u (or i).
            Or if the word ends ç remove the cedilla
And finally:
    Turn a~, o~ back into ã, õ
```

Chapter 16

Russian stemming algorithm

16.1 The stemming algorithm

i-suffixes of Russian tend to be quite regular, with irregularities of declension involving a change to the stem. Irregular forms therefore usually just generate two or more possible stems. Stems in Russian can be very short, and many of the suffixes are also particle words that make ‘natural stopwords’, so a tempting way of running the stemmer is to set a minimum stem length of zero, and thereby reduce to null all words which are made up entirely of suffix parts. We have been a little more cautious, and have insisted that a minimum stem contains one vowel.

The 32 letters of the Russian alphabet are as follows, with the transliterated forms that we will use here shown in brackets:

а (***a***)	б (***b***)	в (***v***)	г (***g***)	д (***d***)	е (***e***)	ж (***zh***)	з (***z***)
и (***i***)	й (***ĭ***)	к (***k***)	л (***l***)	м (***m***)	н (***n***)	о (***o***)	п (***p***)
р (***r***)	с (***s***)	т (***t***)	у (***u***)	ф (***f***)	х (***kh***)	ц (***ts***)	ч (***ch***)
ш (***sh***)	щ (***shch***)	ъ (***”***)	ы (***y***)	ь (***′***)	э (***è***)	ю (***iu***)	я (***ia***)

There is a 33rd letter, ё (***e”***), but it is rarely used and often replaced by е in informal writing. The original algorithm here assumed it had already been mapped to е (***e***); since 2018-03-16 the Snowball implementation we provide performs this mapping for you.

The following are vowels:

а (***a***) е (***e***) и (***i***) о (***o***) у (***u***) ы (***y***) э (***è***) ю (***iu***) я (***ia***)

In any word, *RV* is the region after the first vowel, or the end of the word if it contains no vowel.

R1 is the region after the first non-vowel following a vowel, or the end of the word if there is no such non-vowel.

R2 is the region after the first non-vowel following a vowel in *R1*, or the end of the word if there is no such non-vowel.

For example:

```
p r o t i v o e s t e s t v e n n o m
     |<------       RV        ------>|
       |<-----       R1       ------>|
           |<-----     R2     ------>|
```

(See note on *R1* and *R2*.)

We now define the following classes of ending:

PERFECTIVE GERUND:

group 1: в (***v***) вши (***vshi***) вшись (***vshis'***)

group 2: ив (***iv***) ивши (***ivshi***) ившись (***ivshis'***) ыв (***yv***) ывши (***yvshi***) ывшись (***yvshis'***)

group 1 endings must follow а (***a***) or я (***ia***)

ADJECTIVE:

ее (***ee***) ие (***ie***) ые (***ye***) ое (***oe***) ими (***imi***) ыми (***ymi***) ей (***eì***) ий (***iì***) ый (***yì***) ой (***oì***) ем (***em***) им (***im***) ым (***ym***) ом (***om***) его (***ego***) ого (***ogo***) ему (***emu***) ому (***omu***) их (***ikh***) ых (***ykh***) ую (***uiu***) юю (***iuiu***) ая (***aia***) яя (***iaia***) ою (***oiu***) ею (***eiu***)

PARTICIPLE:

group 1: ем (***em***) нн (***nn***) вш (***vsh***) ющ (***iushch***) щ (***shch***)

group 2: ивш (***ivsh***) ывш (***yvsh***) ующ (***uiushch***)

group 1 endings must follow а (***a***) or я (***ia***)

REFLEXIVE:

ся (***sia***) сь (***s'***)

VERB:

group 1: ла (***la***) на (***na***) ете (***ete***) йте (***ìte***) ли (***li***) й (***ì***) л (***l***) ем (***em***) н (***n***) ло (***lo***) но (***no***) ет (***et***) ют (***iut***) ны (***ny***) ть (***t'***) ешь (***esh'***) нно (***nno***)

group 2: ила (***ila***) ыла (***yla***) ена (***ena***) ейте (***eìte***) уйте (***uìte***) ите (***ite***) или (***ili***) ыли (***yli***) ей (***eì***) уй (***uì***) ил (***il***) ыл (***yl***) им (***im***) ым (***ym***) ен (***en***) ило (***ilo***) ыло (***ylo***) ено (***eno***) ят (***iat***) ует (***uet***) уют (***uiut***) ит (***it***) ыт (***yt***) ены (***eny***) ить (***it'***) ыть (***yt'***) ишь (***ish'***) ую (***uiu***) ю (***iu***)

group 1 endings must follow а (***a***) or я (***ia***)

NOUN:

а (***a***) ев (***ev***) ов (***ov***) ие (***ie***) ье (***'e***) е (***e***) иями (***iiami***) ями (***iami***) ами (***ami***) еи (***ei***) ии (***ii***) и (***i***) ией (***ieì***) ей (***eì***) ой (***oì***) ий (***iì***) й (***ì***) иям (***iiam***) ям (***iam***) ием (***iem***) ем (***em***) ам (***am***) ом (***om***) о (***o***) у (***u***) ах (***akh***) иях (***iiakh***) ях (***iakh***) ы (***y***) ь (***'***) ию (***iiu***) ью (***'iu***) ю (***iu***) ия (***iia***) ья (***'ia***) я (***ia***)

SUPERLATIVE:

ейш (***eìsh***) ейше (***eìshe***)

These are all *i*-suffixes. The list of *d*-suffixes is very short,

DERIVATIONAL:

ост (***ost***) ость (***ost'***)

Define an ADJECTIVAL ending as an ADJECTIVE ending optionally preceded by a PARTICIPLE ending.

For example, in

бегавшая	=	бега	+	вш	+	ая
(*begavshaia*	=	*bega*	+	***vsh***	+	***aia***)

ая (***aia***) is an adjective ending, and вш (***vsh***) a participle ending of group 1 (preceded by the final a (***a***) of бега (***bega***)), so вшая (***vshaia***) is an adjectival ending.

In searching for an ending in a class, always choose the longest one from the class.

So in seaching for a NOUN ending for величие (*velichie*), choose ие (***ie***) rather than е (***e***).

Undouble н (***n***) means, if the word ends нн (***nn***), remove the last letter.

Here now are the stemming rules.

All tests take place in the *RV* part of the word.

So in the test for perfective gerund, the а (***a***) or я (***ia***) which the group 1 endings must follow must itself be in *RV*. In other words the letters before the *RV* region are never examined in the stemming process.

Do each of steps 1, 2, 3 and 4.

Step 1: Search for a PERFECTIVE GERUND ending. If one is found remove it, and that is then the end of step 1. Otherwise try and remove a REFLEXIVE ending, and then search in turn for (1) an ADJECTIVAL, (2) a VERB or (3) a NOUN ending. As soon as one of the endings (1) to (3) is found remove it, and terminate step 1.

Step 2: If the word ends with и (***i***), remove it.

Step 3: Search for a DERIVATIONAL ending in *R2* (i.e. the entire ending must lie in *R2*), and if one is found, remove it.

Step 4: (1) Undouble н (***n***), or, (2) if the word ends with a SUPERLATIVE ending, remove it and undouble н (***n***), or (3) if the word ends ь (**'**) (soft sign) remove it.

Chapter 17

Spanish stemming algorithm

17.1 Design Notes

Accents are often missing in informally written Spanish. Since Snowball 3.0.0 some additional rules are included to try to handle commonly occurring cases where accents are omitted and an additional rule doesn't cause problems with other words. There's likely scope for further such improvements - please report instances.

17.2 History of functional changes to the algorithm

- 2005-06-15: Added rules to remove *-ante*, *-antes*, *-ancia* and *-ancias*
- Snowball 3.0.0: Added rules to remove *-acion* and *-ucion*.

17.3 The stemming algorithm

Letters in Spanish include the following accented forms,

á é í ó ú ü ñ

The following letters are vowels:

a e i o u á é í ó ú ü

R2 is defined in the usual way — see the note on *R1* and *R2*.

RV is defined as follows (and this is not the same as the French stemmer definition):

If the second letter is a consonant, *RV* is the region after the next following vowel, or if the first two letters are vowels, *RV* is the region after the next consonant, and otherwise (consonant-vowel case) *RV* is the region after the third letter. But *RV* is the end of the word if these positions cannot be found.

For example,

```
m a c h o     o l i v a     t r a b a j o     á u r e o
      |...|         |...|         |.......|         |...|
```

Always do steps 0 and 1.

Step 0: Attached pronoun

Search for the longest among the following suffixes

me se sela selo selas selos la le lo las les los nos

and delete it, if comes after one of

(*a*) ***iéndo ándo ár ér ír***
(*b*) ***ando iendo ar er ir***
(*c*) ***yendo*** following ***u***

in *RV*. In the case of (*c*), ***yendo*** must lie in *RV*, but the preceding ***u*** can be outside it.

In the case of (*a*), deletion is followed by removing the acute accent (for example, *haciéndola* » *haciendo*).

Step 1: Standard suffix removal

Search for the longest among the following suffixes, and perform the action indicated.

anza anzas ico ica icos icas ismo ismos able ables ible ibles ista istas oso osa osos osas amiento amientos imiento imientos

delete if in *R2*

adora ador ación adoras adores aciones ante antes ancia ancias acion

delete if in *R2*

if preceded by ***ic***, delete if in *R2*

logía logías

replace with ***log*** if in *R2*

ución uciones ucion

replace with ***u*** if in *R2*

encia encias

replace with ***ente*** if in *R2*

amente

delete if in *R1*

if preceded by ***iv***, delete if in *R2* (and if further preceded by ***at***, delete if in *R2*), otherwise,

if preceded by ***os***, ***ic*** or ***ad***, delete if in *R2*

mente

delete if in *R2*

if preceded by ***ante***, ***able*** or ***ible***, delete if in *R2*

idad idades

delete if in *R2*

if preceded by ***abil***, ***ic*** or ***iv***, delete if in *R2*

iva ivo ivas ivos

delete if in *R2*

if preceded by ***at***, delete if in *R2*

Do step *2a* if no ending was removed by step 1.

Step *2a*: Verb suffixes beginning ***y***

Search for the longest among the following suffixes in *RV*, and if found, delete if preceded by ***u***.

ya ye yan yen yeron yendo yo yó yas yes yais yamos

(Note that the preceding u need not be in *RV*.)

Do Step *2b* if step *2a* was done, but failed to remove a suffix.

Step *2b*: Other verb suffixes

Search for the longest among the following suffixes in *RV*, and perform the action indicated.

en es éis emos

delete, and if preceded by ***gu*** delete the ***u*** (the ***gu*** need not be in *RV*)

arían arías arán arás aríais aría aréis aríamos aremos ará aré erían erías erán erás eríais ería eréis eríamos eremos erá eré irían irías irán irás iríais iría iréis iríamos iremos irá iré aba ada ida ía ara iera ad ed id ase iese aste iste an aban ían aran ieran asen iesen aron ieron ado ido ando iendo ió ar er ir as abas adas idas ías aras ieras ases ieses ís áis abais íais arais ierais aseis ieseis asteis isteis ados idos amos ábamos íamos imos áramos iéramos iésemos ásemos

delete

Always do step 3.

Step 3: residual suffix

Search for the longest among the following suffixes in *RV*, and perform the action indicated.

os a o á í ó

delete if in *RV*

e é

delete if in *RV*, and if preceded by ***gu*** with the ***u*** in *RV* delete the ***u***

And finally:

Remove acute accents

Chapter 18

Swedish stemming algorithm

18.1 The stemming algorithm

The Swedish alphabet includes the following additional letters,

ä å ö

The following letters are vowels:

a e i o u y ä å ö

R2 is not used: *R1* is defined in the same way as in the German stemmer. (See the note on *R1* and *R2*.)

Define a valid ***s***-ending as one of

b c d f g h j k l m n o p r t v y

Define a valid ***öst***-ending as one of

i k l n p r t u v

Define a valid ***et***-ending as at least one letter followed by a vowel followed by a non-vowel, which does not have one of the following as a suffix

h iet uit fab cit dit alit ilit mit nit pit rit sit tit ivit kvit xit kom rak pak stak

Do each of steps 1, 2 and 3.

Step 1:

Search for the longest among the following suffixes in *R1*, and perform the action indicated.

(*a*) ***a arna erna heterna orna ad e ade ande arne are aste en anden aren heten ern ar er heter or as arnas ernas ornas es ades andes ens arens hetens erns at andet het ast***

delete

(*b*) ***s***

if preceded by ***et*** and that is preceded by a valid ***et***-ending remove both ***s*** and ***et***, otherwise delete if preceded by a valid ***s***-ending

(*c*) ***et***

delete if preceded by a valid ***et***-ending

(Note that only the suffix needs to be in *R1*, the letter(s) of the valid ***s***-ending or ***et***-ending are not required to be.)

Step 2:

Search for one of the following suffixes in *R1*, and if found delete the last letter.

dd gd nn dt gt kt tt

(For example, *friskt* » *frisk*, *fröknarnn fröknarn*)

Step 3:

Search for the longest among the following suffixes in *R1*, and perform the action indicated.

lig ig els

delete

öst

replace with ***ös*** if preceded by a valid ***öst***-ending

fullt

replace with ***full***

(The letter of the valid ***öst***-ending is not necessarily in *R1*. Prior to Snowball 3.0.0, ***öst***-ending was effectively just ***l*** and *was* required to be in *R1*.)

18.2 Design Notes

Swedish has a noun ending corresponding to the definite article (*the* in English). This occurs very commonly but cannot always be removed with certainty. Currently the algorithm removes the ***en*** form, and the ***et*** form in some cases, but not the ***t*** or ***n*** forms,

husen		hus
valet		val
flickan	»	flickan
äpplet		äpplet

18.3 History of functional changes to the algorithm

- Snowball 3.0.0: Change ***öst*** suffix to ***ös*** in more situations.
- Snowball 3.0.0: New rule to remove some ***et*** suffixes.

Part III

Index

Glossary

a-suffix

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.

i-suffix

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)

d-suffix

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

Indo-European languages

Most European and many Asian languages belong to the Indo-European language group. Historically, it includes the Latin, Greek, Persian and Sanskrit of the ancient world, and with the rise of the European empires, languages of this group are now dominant in the Americas, Australia and large parts of Africa. Indo-European languages are therefore the main languages of modern Western culture, and they are all similarly amenable to stemming.

The Indo-European group has many recognisable sub-groups, for example *Romance* (Italian, French, Spanish ...), *Slavonic* (Russian, Polish, Czech ...), *Celtic* (Irish Gaelic, Scottish Gaelic, Welsh ...). The *Germanic* sub-group includes German and Dutch, and the *Scandinavian* languages are also usually classed as Germanic, although for convenience we have made a separate grouping of them on the Snowball site. English is also classed as Germanic, although it has been classed separately by us. This is not for reasons of narrow chauvinism, but because the suffix structure of English clearly lies mid-way between the Germanic and Romance groups, and it therefore requires separate treatment.

Uralic languages

The Uralic languages are spoken mainly in Northern Russia and Europe. They are divided into Samoyed, spoken mainly in the Siberian region, and Finno-Ugric, spoken mainly in Europe. Although the number of languages in the group is substantial, the total number of speakers is relatively small. The best known Uralic languages are perhaps Hungarian, Finnish and Estonian. Finnish and Estonian are in fact fairly similar. On the other hand Hungarian and Finnish are as different as are, say, French and Persian in the Indo-European group.

Like the Indo-European languages, the Uralic languages are amenable to stemming.

License

Except where explicitly noted, all the software given out on this Snowball site is covered by the 3-clause BSD License:

Essentially, all this means is that you can do what you like with the code, except claim another Copyright for it, or claim that it is issued under a different license. The software is also issued without warranties, which means that if anyone suffers through its use, they cannot come back and sue you. You also have to alert anyone to whom you give the Snowball software to the fact that it is covered by the BSD license.

Credits

Snowball, and most of the current stemming algorithms were written by Dr Martin Porter, who also prepared the material for the Website. The Snowball to Java code generator, and supporting Java libraries, were contributed by Richard Boulton. Dr Andrew MacFarlane, of City University, London, gave much initial encouragement and proofreading assistance.

Richard Boulton established the original Snowball website, from which this website has evolved.

Linguistic assistance for Russian, German and Dutch has been provided by Patrick Miles (of the Patrick Miles Translation Agency, Cambridge, UK). Pat is a distinguished translator, whose English versions of Chekhov have appeared on the London stage.

Various emailers have helped improve the stemmers with their many suggestions and comments. We must especially mention Andrei Aksyonoff and Oleg Bartunov (Russian), Steve Tolkin and Wendy Reetz (English), and Fred Brault (French). Blake Madden found a number of elusive errors in the stemmer descriptions.

Anna Tordai has provided the Hungarian stemming algorithm.

Evren (Kapusuz) Cilden has provided the Turkish stemming algorithm.

Olly Betts has made a significant performance improvement to the C code generator.

The Snowball mailing lists are hosted for us free by James Aylett, who owns and runs the machine that hosts the tartarus website.

We received two Romanian stemming algorithms in 2006, from Erwin Glockner, Doina Gliga and Marina Stegarescu, working at Heidelberg, and from Irina Tirdea in Bucharest. After some experimentation, the Snowball Romanian stemmer has been rewritten from scratch, but the basic list of verb endings with their separation into two groups with different removal criteria is taken from Irina Tirdea's stemmer.

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.