The Porter Stemmer

SUSHMITHA KATAM

Indiana State University

skatam@sycamores.indstate.edu

December 6, 2014

Abstract

This paper summarises the main features of the algorithm, and highlights its role not just in modern information retrieval research, but also in a range of related subject domains.Removing suffixes by automatic means is an operation which is especially useful in the field of information retrieval. In a typical IR environment, one has a collection of documents, each described by the words in the document title and possibly by words in the document abstract. Ignoring the issue of precisely where the words originate, we can say that a document is represented by a vector of words, or Terms with a common stem will usually have similar meanings, for example:

CONNECT

CONNECTED

CONNECTING

CONNECTION

CONNECTIONS

Frequently, the performance of an IR system will be improved if term groups such as this are conflated into a single term. This may be done by removal of the various suffixes -ED, -ING, -ION, IONS to leave the single term CONNECT. In addition, the suffix stripping process will reduce the total number of terms in the IR system, and hence reduce the size and complexity of the data in the system, which is always advantageous.

1 Introduction

Natural language texts typically contain many different variants of a basic word. Morphological variants (e.g., COMPUTATIONAL, COMPUTER, COMPUTERS, COMPUTING etc.) are generally the most common, with other sources including valid alternative spellings, misspellings, and variants arising from transliteration and abbreviation. The effectiveness of searching, most obviously but not exclusively in terms of recall, would be expected to increase if it were possible to conflate (i.e., to bring together) the variants of a given word so that they could all be retrieved in response to a query that specified just a single variant.

Lovins,described the first stemmer to be developed specifically for information-retrieval applications and introduced the idea of stemming based on a dictionary of common suffixes,such as *SES, *ING or *ATION. This algorithm spurred the development of many subsequent algorithms.When a word is presented for stemming in a dictionary-based stemming algorithm, the

right hand end of the word is checked for the presence of any of the suffixes in the dictionary. If a suffix is found to be present, it is removed, subject to a range of context-sensitive rules that forbid, e.g., the removal of *ABLE from TABLE or of *S from GAS;

- approach Conflates inflected/derived words to a stem (root)
- Words like Abate, abated, abatement, abatements, abates might all stem to .abat. .Stemmers might produce different stems any suffix stripping program for IR work, two points must be borne in mind. Firstly, the suffixes are being removed simply to improve IR performance, and not as a linguistic exercise.
- second point is that with the approach adopted here, i.e. the use of a suffix list with various rules, the success rate for the suffix stripping will be significantly less than 100evaluated.

1.1 Common Features

Historically, the following shortcomings have been found in other encodings of the stemming algorithm.

The algorithm clearly explains that when a set of rules of the type (condition)S1 . S2 are presented together, only one rule is applied, the one with the longest matching suffix S1 for the given word. This is true whether the rule succeeds or fails (i.e. whether or not S2 replaces S1). Despite this, the rules are sometimes simply applied in turn until either one of them succeeds or the list runs out.

This leads to small errors in various places, for example

(m greater than 1)ement .

(m greater than 1)ment .

(m greater than 1)ent .

to remove final ement, ment and ent.

Properly, argument stems to argument. The longest matching suffix is -ment. Then stem argu- has measure m equal to 1 and so -ment will not be removed. But if the three rules are applied in turn, then for suffix -ent the stem argum- has measure m equal to 2, and -ent gets removed.

The more delicate rules are liable to misinterpretation. (Perhaps greater care was required in explaining them.) So

((m greater than 1) and (*s or *t))ion is taken to mean

(m greater than 1)(s or t)ion The former means that taking off -ion leaves a stem with measure greater than 1 ending -s or -t; the latter means that taking off -sion or -tion leaves a stem of measure greater than 1. A similar confusion tends to arise in interpreting rule 5b, to reduce final double L to single L.

Occasionally cruder errors have been seen. For example the test for Y being consonant or vowel set up the wrong way round.

It is interesting that although the published paper explains how to do the tests on the strings S1 by a program switch on the last or last but one letter, many encodings fail to use this technique, making them much slower than they need be.

2 The Algorithm

consonant in a word is a letter other than A, E, I, O or U, and other than Y preceded by a consonant. (The fact that the term 'consonant' is defined to some extent in terms of itself does

not make it ambiguous.) So in TOY the consonants are T and Y, and in SYZYGY they are S, Z and G. If a letter is not a consonant it is a vowel.

A consonant will be denoted by c, a vowel by v. A list ccc... of length greater than 0 will be denoted by C, and a list vvv... of length greater than 0 will be denoted by V. Any word, or part of a word, therefore has one of the four forms:

CVCV ... C

CVCV ... V

VCVC ... C

VCVC ... V

These may all be represented by the single form

C VCVC ... [V]

where the square brackets denote arbitrary presence of their contents. Using (VC)m to denote VC repeated m times, this may again be written as

C (VC)m[V].

m will be called the of any word or word part when represented in this form. The case m = 0 covers the null word. Here are some examples:

m=0 TR, EE, TREE, Y, BY.

m=1 TROUBLE, OATS, TREES, IVY.

m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

The rules for removing a suffix will be given in the form

(condition) S1 S2

This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of m, e.g.

(m greater than 1) EMENT

Here S1 is 'EMENT' and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which m = 2.

The 'condition' part may also contain the following:

*S - the stem ends with S (and similarly for the other letters).

v - the stem contains a vowel.

*d - the stem ends with a double consonant (e.g. -TT, -SS).

*o - the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP).

And the condition part may also contain expressions with and, or, and not, so that

(m greater than 1 and (*S or *T))

tests for a stem with m greater than 1 ending in S or T, while

(*d and not (*L or *S or *Z))

tests for a stem ending with a double consonant other than L, S or Z. Elaborate conditions like this are required only rarely.

In a set of rules written beneath each other, only one is obeyed, and this will be the one with the longest matching S1 for the given word. For example, with

SSES SS

IES I

SS SS

S

(here the conditions are all null) CARESSES maps to CARESS since SSES is the longest match for S1. Equally CARESS maps to CARESS (S1='SS') and CARES to CARE (S1='S').

In the rules below, examples of their application, successful or otherwise, are given on the right in lower case. The algorithm now follows:

Step 1a

SSES SS caresses caress

IES I ponies poni

ties ti

SS SS caress caress

S cats cat

Step 1b

(m greater than 0) EED EE feed feed agreed agree

(*v*) ED plastered plaster bled bled

(*v*) ING motoring motor sing sing

If the second or third of the rules in Step 1b is successful, the following is done:

AT ATE conflat(ed) conflate

BL BLE troubl(ed) trouble

IZ IZE siz(ed) size

(*d and not (*L or *S or *Z))

single letter

hopp(ing) hop

tann(ed) tan

fall(ing) fall

hiss(ing) hiss

fizz(ed) fizz

fail(ing) fail

fil(ing) file

The rule to map to a single letter causes the removal of one of the double letter pair. The -E is put back on -AT, -BL and -IZ, so that the suffixes -ATE, -BLE and -IZE can be recognised later. This E may be removed in step 4.

Step 1c

(*v*) Y I

happy happi

sky sky

Step 1 deals with plurals and past participles. The subsequent steps are much more straightforward.

Step 2

(m greater than 0) ATIONAL ATE relational relate

(m greater than 0) TIONAL TION conditional condition rational rational

(m greater than 0) ENCI ENCE valenci valence

(m greater than 0) ANCI ANCE hesitanci hesitance

(m greater than 0) IZER IZE digitizer digitize

(m greater than 0) ABLI ABLE conformabli conformable

(m greater than 0) ALLI AL radicalli radical

(m greater than 0) ENTLI ENT differentli different

(m greater than 0) ELI E vileli vile

(m greater than 0) OUSLI OUS analogousli analogous

(m greater than 0) IZATION IZE vietnamization vietnamize

(m greater than 0) ATION ATE predication predicate

(m greater than 0) ATOR ATE operator operate

(m greater than 0) ALISM AL feudalism feudal

(m greater than 0) IVENESS IVE decisiveness decisive

(m greater than 0) FULNESS FUL hopefulness hopeful

(m greater than 0) OUSNESS OUS callousness callous

(m greater than 0) ALITI AL formaliti formal

(m greater than 0) IVITI IVE sensitiviti sensitive

(m greater than 0) BILITI BLE sensibiliti sensible

The test for the string S1 can be made fast by doing a program switch on the penultimate letter of the word being tested. This gives a fairly even breakdown of the possible values of the string S1. It will be seen in fact that the S1-strings in step 2 are presented here in the alphabetical order of their penultimate letter. Similar techniques may be applied in the other steps.

Step 3

(m greater than 0) ICATE IC triplicate triplic

(m greater than 0) ATIVE formative form

(m greater than 0) ALIZE AL formalize formal

(m greater than 0) ICITI IC electriciti electric

(m greater than 0) ICAL IC electrical electric

(m greater than 0) FUL hopeful hope

(m greater than 0) NESS goodness good

Step 4

(m greater than 1) AL revival reviv

(m greater than 1) ANCE allowance allow

(m greater than 1) ENCE inference infer

(m greater than 1) ER airliner airlin

(m greater than 1) IC gyroscopic gyroscop

(m greater than 1) ABLE adjustable adjust

(m greater than 1) IBLE defensible defens

(m greater than 1) ANT irritant irrit

(m greater than 1) EMENT replacement replac

(m greater than 1) MENT adjustment adjust

(m greater than 1) ENT dependent depend

(m greater than 1 and (*S or *T)) ION adoption adopt

(m greater than 1) OU homologou homolog

(m greater than 1) ISM communism commun

(m greater than 1) ATE activate activ

(m greater than 1) ITI angulariti angular

(m greater than 1) OUS homologous homolog

(m greater than 1) IVE effective effect

(m greater than 1) IZE bowdlerize bowdler

The suffixes are now removed. All that remains is a little tidying up.

Step 5a

(m greater than 1) E probate probat rate rate

(m=1 and not *o) E cease ceas

Step 5b

(m greater than 1 and *d and *L) single letter controll control roll roll

The algorithm is careful not to remove a suffix when the stem is too short, the length of the stem being given by its measure, m. There is no linguistic basis for this approach. It was merely observed that m could be used quite effectively to help decide whether or not it was wise to take off a suffix.

For example, in the following two lists:

list A list B

RELATE DERIVATE

PROBATE ACTIVATE

CONFLATE DEMONSTRATE

PIRATE NECESSITATE

PRELATE RENOVATE

-ATE is removed from the list B words, but not from the list A words. This means that the pairs DERIVATE or DERIVE, ACTIVATE or ACTIVE, DEMONSTRATE or DEMONSTRABLE, NECESSITATE or NECESSITOUS, will conflate together. The fact that no attempt is made to identify prefixes can make the results look rather inconsistent. Thus PRELATE does not lose the -ATE, but ARCHPRELATE becomes ARCHPREL. In practice this does not matter too much, because the presence of the prefix decreases the probability of an erroneous conflation.

3 Frequently Asked Questions

FAQs (frequently asked questions)

1. What is the licensing arrangement for this software?

   This question has become very popular recently (the period 2008.2009), despite the clear statment above that ..all these encodings of the algorithm can be used free of charge for any purpose... The problem I think is that intellectual property has become such a major issue that some more formal statement is expected. So to restate it:

   The software is completely free for any purpose, unless notes at the head of the program text indicates otherwise (which is rare). In any case, the notes about licensing are never more restrictive than the BSD License.

In every case where the software is not written by me (Martin Porter), this licensing arrangement has been endorsed by the contributor, and it is therefore unnecessary to ask the contributor again to confirm it.

I have not asked any contributors (or their employers, if they have them) for proofs that they have the right to distribute their software in this way.

(For anyone taking software from the Snowball website, the position is similar but simpler. There, all the software is issued under the BSD License, and for contributions not written by Martin Porter and Richard Boulton, we have again not asked the authors, or the authors. employers, for proofs that they have such distribution rights.)

2. Why is the stemmer not producing proper words?

   It is often taken to be a crude error that a stemming algorithm does not leave a real word after removing the stem. But the purpose of stemming is to bring variant forms of a word together, not to map a word onto its .paradigm. form.

   And connected with this,

3. Why are there errors?

   The question normally comes in the form, why should word X be stemmed to x1, when one would have expected it to be stemmed to x2? It is important to remember that the stemming algorithm cannot achieve perfection. On balance it will (or may) improve IR performance, but in individual cases it may sometimes make what are, or what seem to be, errors. Of course, this is a different matter from suggesting an additional rule that might be included in the stemmer to improve its performance.

   /endenumerate

References I will add to this from time to time any noteworthy instructive or educational links. Suggestions welcome.

a) http://people.ischool.berkeley.edu/ hearst/irbook/porter.html
b) http://www.cs.cmu.edu/ callan/Teaching/porter.c
c) http://snowball.tartarus.org/algorithms/porter/stemmer.html
d) http://programmingpraxis.com/2009/09/08/porter-stemming/
e) http://tartarus.org/martin/PorterStemmer/
f) http://www.eecis.udel.edu/ trnka/CISC889-11S/lectures/dan-porters.pdf