This paper has been left in plain text form for anyone who wants to copy the rules out as program comment.

\...\ denotes italicisation; {...} denotes superscripting

An algorithm for suffix stripping

M.F.Porter
1980

Originally published in \Program\, \14\ no. 3, pp 130-137, July 1980. (A few typos have been corrected.)

1. INTRODUCTION

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 vetor of words, or \terms\. 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.

Many strategies for suffix stripping have been reported in the literature.{(e.g. 1-6)} The nature of the task will vary considerably depending on whether a stem dictionary is being used, whether a suffix list is being used, and of course on the purpose for which the suffix stripping is being done. Assuming that one is not making use of a stem dictionary, and that the purpose of the task is to improve IR performance,

the suffix stripping program will usually be given an explicit list of suffixes, and, with each suffix, the criterion under which it may be removed from a word to leave a valid stem. This is the approach adopted here. The main merits of the present program are that it is small (less than 400 lines of BCPL), fast (it will process a vocabulary of 10,000 different words in about 8.1 seconds on the IBM 370/165 at Cambridge University), and reasonably simple. At any rate, it is simple enough to be described in full as an algorithm in this paper. (The present version in BCPL is freely available from the author. BCPL is itself available on a wide range of different computers, but anyone wishing to use the program should have little difficulty in coding it up in other programming languages.) Given the speed of the program, it would be quite realistic to apply it to every word in a large file of continuous text, although for historical reasons we have found it convenient to apply it only to relatively small vocabulary lists derived from continuous text files.

In 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. This means that it would not be at all obvious under what circumstances a suffix should be removed, even if we could exactly determine the suffixes of a word by automatic means.

Perhaps the best criterion for removing suffixes from two words W1 and W2 to produce a single stem S, is to say that we do so if there appears to be no difference between the two statements `a document is about W1' and `a document is about W2'. So if W1=`CONNECTION' and W2=`CONNECTIONS' it seems very reasonable to conflate them to a single stem. But if W1=`RELATE' and W2=`RELATIVITY' it seems perhaps unreasonable, especially if the document collection is concerned with theoretical physics. (It should perhaps be added that RELATE and RELATIVITY \are\ conflated together in the algorithm described here.) Between these two extremes there is a continuum of different cases, and given two terms W1 and W2, there will be some variation in opinion as to whether they should be conflated, just as there is with deciding the relevance of some document to a query. The evaluation of the worth of a suffix stripping system is correspondingly difficult.

The 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 100% irrespective of how the process is

evaluated. For example, if SAND and SANDER get conflated, so most probably
will WAND and WANDER. The error here is that the -ER of WANDER has been treated as a suffix when in fact it is part of the stem. Equally, a suffix
may completely alter the meaning of a word, in which case its removal is unhelpful. PROBE and PROBATE for example, have quite distinct meanings in modern English. (In fact these would not be conflated in our present algorithm.) There comes a stage in the development of a suffix stripping program where the addition of more rules to increase the performance in one
area of the vocabulary causes an equal degradation of performance elsewhere.
Unless this phenomenon is noticed in time, it is very easy for the program
to become much more complex than is really necessary. It is also easy to give undue emphasis to cases which appear to be important, but which turn ut
to be rather rare. For example, cases in which the root of a word changes with the addition of a suffix, as in DECEIVE/DECEPTION, RESUME/RESUMPTION,
INDEX/INDICES occur much more rarely in real vocabularies than one might at
first suppose. In view of the error rate that must in any case be expected,
it did not seem worthwhile to try and cope with these cases.

It is not obvious that the simplicity of the present program is any demerit.
In a test on the well-known Cranfield 200 collection{7} it gave an improvement in retrieval performance when compared with a very much more elaborate program which has been in use in IR research in Cambridge since 1971{(2,6)}. The test was done as follows: the words of the titles and abstracts in the documents were passed through the earlier suffix stripping
system, and the resultis stems were used to index the documents. The words
of the queries were reduced to stems in the same way, and the documents were
ranked for each query using term coordination matching of query against document. From these rankings, recall and precision values were obtained using the standard recall cutoff method. The entire process was then repeated using the suffix stripping system described in this paper, and the
results were as follows:

earlier system		present system	
precision	recall	precision	recall
0	57.24	0	58.60
10	56.85	10	58.13
20	52.85	20	53.92
30	42.61	30	43.51
40	42.20	40	39.39

50	39.06	50	38.85
60	32.86	60	33.18
70	31.64	70	31.19
80	27.15	80	27.52
90	24.59	90	25.85
100	24.59	100	25.85

Cleary, the performance is not very different. The important point is that
the earlier, more elaborate system certainly performs no better than the present, simple system.

(This test was done by prof. C.J. van Rijsbergen.)

2. THE ALGORITHM

To present the suffix stripping algorithm in its entirety we will need a few
difinitions.

A \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 \measure\ 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 > 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>1 and (*S or *T))

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

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

tests for a stem ending witha 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>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
(m=1 and *o)	-> E	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>0) ATIONAL	-> ATE	relational	->	relate

(m>0) TIONAL	->	TION	conditional	->	condition
			rational	->	rational
(m>0) ENCI	->	ENCE	valenci	->	valence
(m>0) ANCI	->	ANCE	hesitanci	->	hesitance
(m>0) IZER	->	IZE	digitizer	->	digitize
(m>0) ABLI	->	ABLE	conformabli	->	conformable
(m>0) ALLI	->	AL	radicalli	->	radical
(m>0) ENTLI	->	ENT	differentli	->	different
(m>0) ELI	->	E	vileli	- >	vile
(m>0) OUSLI	->	OUS	analogousli	->	analogous
(m>0) IZATION	->	IZE	vietnamization	->	vietnamize
(m>0) ATION	->	ATE	predication	->	predicate
(m>0) ATOR	->	ATE	operator	->	operate
(m>0) ALISM	->	AL	feudalism	->	feudal
(m>0) IVENESS	->	IVE	decisiveness	->	decisive
(m>0) FULNESS	->	FUL	hopefulness	->	hopeful
(m>0) OUSNESS	->	OUS	callousness	->	callous
(m>0) ALITI	->	AL	formaliti	->	formal
(m>0) IVITI	->	IVE	sensitiviti	->	sensitive
(m>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>0) ICATE	->	IC	triplicate	->	triplic
(m>0) ATIVE	->		formative	->	form
(m>0) ALIZE	->	AL	formalize	->	formal
(m>0) ICITI	->	IC	electriciti	->	electric
(m>0) ICAL	->	IC	electrical	->	electric
(m>0) FUL	->		hopeful	->	hope
(m>0) NESS	->		goodness	->	good

Step 4

(m>1) AL	->		revival	->	reviv
(m>1) ANCE	->		allowance	->	allow
(m>1) ENCE	->		inference	->	infer
(m>1) ER	->		airliner	->	airlin
(m>1) IC	->		gyroscopic	->	gyroscop
(m>1) ABLE	->		adjustable	->	adjust
(m>1) IBLE	->		defensible	->	defens
(m>1) ANT	->		irritant	->	irrit
(m>1) EMENT	->		replacement	->	replac
(m>1) MENT	->		adjustment	->	adjust
(m>1) ENT	->		dependent	->	depend

```
    (m>1 and (*S or *T)) ION ->     adoption       ->  adopt
    (m>1) OU    ->                  homologou      ->  homolog
    (m>1) ISM   ->                  communism      ->  commun
    (m>1) ATE   ->                  activate       ->  activ
    (m>1) ITI   ->                  angulariti     ->  angular
    (m>1) OUS   ->                  homologous     ->  homolog
    (m>1) IVE   ->                  effective      ->  effect
    (m>1) IZE   ->                  bowdlerize     ->  bowdler
```

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

Step 5a

```
    (m>1) E     ->                  probate        ->  probat
                                    rate           ->  rate
    (m=1 and not *o) E ->           cease          ->  ceas
```

Step 5b

```
    (m > 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/DERIVE, ACTIVATE/ACTIVE, DEMONSTRATE/DEMONS-
TRABLE, NECESSITATE/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.

Complex suffixes are removed bit by bit in the different steps. Thus

GENERALIZATIONS is stripped to GENERALIZATION (Step 1), then to GENERALIZE
(Step 2), then to GENERAL (Step 3), and then to GENER (Step 4). OSCILLATORS
is stripped to OSCILLATOR (Step 1), then to OSCILLATE (Step 2), then to OSCILL (Step 4), and then to OSCIL (Step 5). In a vocabulary of 10,000 words, the reduction in size of the stem was distributed among the steps as
follows:

Suffix stripping of a vocabulary of 10,000 words	
Number of words reduced in step 1:	3597
" 2:	766
" 3:	327
" 4:	2424
" 5:	1373
Number of words not reduced:	3650

The resulting vocabulary of stems contained 6370 distinct entries. Thus the
suffix stripping process reduced the size of the vocabulary by about one third.

REFERENCIES

1. LOVINS, J.B. Development of a Stemming Algorithm. \Mechanical Translation and computation Linguistics\. \11\ (1) March 1968 pp 23-31.

2. ANDREWS, K. The Development of a Fast Conflation Algorithm for English.
\Dissertation for the Diploma in Computer Science\, Computer Laboratory, University of Cambridge, 1971.

3. PETRARCA, A.E. and LAY W.M. Use of an automatically generated authority
list to eliminate scattering caused by some singular and plural main index terms. \Proceedings of the American Society for Information Science\, \6\ 1969 pp 277-282.

4. DATTOLA, Robert T. \FIRST: Flexible Information Retrieval System for Text\. Webster N.Y: Xerox Corporation, 12 Dec 1975.

5. COLOMBO, D.S. and NIEHOFF R.T. \Final report on improved access to scientific and technical information through automated vocabulary switching.\ NSF Grant No. SIS75-12924 to the National Science Foundation.

6. DAWSON, J.L. Suffix Removal and Word Conflation. \ALLC Bulletin\, Michaelmas 1974 p.33-46.

7. CLEVERDON, C.W., MILLS J. and KEEN M. \Factors Determining the Performance of Indexing Systems\ 2 vols. College of Aeronautics,

Cranfield 1966.