Stemming

Sometimes all you care about is the stem (or root) of a word:

- **walk**
- **walk**s
- **walk**ing
- **walk**er
- **walk**ed

Motivation: topic modelling

vs. lemmatization

lemma: The canonical form of an inflected word; i.e., the form usually found as the headword in a dictionary, such as the nominative singular of a noun, the bare infinitive of a verb, etc.

-- Wiktionary

"George began to walk and continued walking until he walked into a wall."

What is this about?

- George
- began
- walk
- continued
- walking
- walked
- wall

walk, walks, walking, walked → walk

continue, continues, continued, continuing, continuity → continu

wall, walled, walling → wall

vs. lemmatization

walk, walks, walked, walking, walker → to walk

continue, continues, continued, continuing, continuity → to continue

be, am, are, is, was, were, been, being → to be

Porter stemmer

Martin Porter 1979-2006

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').

```
1a. SSES->SS
    IES->I
    ...
1b. (*v*)ED->
    (*v*)ING->
    ...
2.  (m>0)ATION->ATE
    (m>0)INENESS->IVE
    ...
3.  (m>0)ICATE->IC
    (m>0)ATIVE->
    ...
4.   ...
5.   ...
```

Porter stemmer: shortcomings

- entirely heuristic, but English is messy:
  brought ->! bring
  hung ->! hang
- machine learning can solve many of these harder of problems
- the lost information is frequently important
- better to dissect the word entirely, e.g. Luong 2013

Alternatives

Luong 2013