LING/C SC/PSYC 438/538

Lecture 26
Sandiway Fong
Today's Topics

• Optional Homework 14 graded
• **Reminder**: 538 Presentations start next time (*see next slide*)
• Last topic of the semester: **Morphology and Stemming**
  • Chapter 3 of JM
    • Morphology: Introduction
    • Stemming

stemming, 68, 772
and morphological parsing, 46
Porter stemmer, 46
sophisticated methods, 772
## 538 Presentations

<table>
<thead>
<tr>
<th>Name</th>
<th>Topic</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew Zupon</td>
<td>20.7 Word Similarity: Distributional Methods</td>
<td>30-Nov</td>
</tr>
<tr>
<td>Maria Alexeeva</td>
<td>24.1-2 Properties of Human Conversation + Basic Dialogue Systems</td>
<td>30-Nov</td>
</tr>
<tr>
<td>Roya Kabiri</td>
<td>19.1-4 Word Senses -&gt; Event Participants</td>
<td>30-Nov</td>
</tr>
<tr>
<td>Seongjin Park</td>
<td>17.1-2 Computational Desiderata for Representation + Model-Theoretic Semantics</td>
<td>30-Nov</td>
</tr>
<tr>
<td>John Culnan</td>
<td>23.3-4 Summarization and Multi-text summarization</td>
<td>30-Nov</td>
</tr>
<tr>
<td>Chen Chen</td>
<td>18.1-3 Computational Semantics</td>
<td>30-Nov</td>
</tr>
<tr>
<td>Jeffrey Chiavetta</td>
<td>15.1-3 Feature Structures</td>
<td>30-Nov</td>
</tr>
<tr>
<td>Jiaheng Xie</td>
<td>22.1 Named Entity Recognition</td>
<td>5-Dec</td>
</tr>
<tr>
<td>Brendan Murphy</td>
<td>25.5-6 Alignment in MT + Training Alignment Models</td>
<td>5-Dec</td>
</tr>
<tr>
<td>Mohammadreza Ebrahim</td>
<td>22.3 Temporal and Event Processing</td>
<td>5-Dec</td>
</tr>
<tr>
<td>Rebecca Cordes</td>
<td>17.3-4 First Order Logic</td>
<td>5-Dec</td>
</tr>
<tr>
<td>Rianne Lyons</td>
<td>24.7 Plan-Based Dialog Agents</td>
<td>5-Dec</td>
</tr>
<tr>
<td>Yiyun Zhao</td>
<td>20.1-5 Word Sense Disambiguation</td>
<td>5-Dec</td>
</tr>
<tr>
<td>Ragheb Al-Ghezi</td>
<td>21.7-8 Coference Resolution and Evaluation</td>
<td>5-Dec</td>
</tr>
<tr>
<td>Sandeep Suntwal</td>
<td>20.5-6 Minimally Supervised WSD + Thesaurus Methods</td>
<td>5-Dec</td>
</tr>
</tbody>
</table>
Morphology

• **Inflectional Morphology:**
  • *basically: no change in category*

  • *ϕ-features* (person, number, gender)
    • Examples: movies, blonde, actress
    • Irregular examples:
      • appendices (from appendix), geese (from goose)

  • **case**
    • Examples: he/him, who/whom

  • **comparatives and superlatives**
    • Examples: happier/happiest

  • **tense**
    • Examples: drive/drives/drove (-ed)/driven

• **Derivational Morphology**
  • *basically: category changing*

  • **nominalization**
    • Examples: formalization, informant, informer, refusal, lossage

  • **deadjectivals**
    • Examples: weaken, happiness, simplify, formalize, slowly, calm

  • **deverbals**
    • Examples: see nominalization, readable, employee

  • **denominals**
    • Examples: formal, bridge, ski, cowardly, useful
Morphology and Semantics

• **Morphemes**: *units of meaning*

• **suffixation**
  - Examples:
    - $x$ employ $y$
      - employee: picks out $y$
      - employer: picks out $x$
    - $x$ read $y$
      - readable: picks out $y$

• **prefixation**
  - Examples:
    - undo, redo, un-redo, en-code, de-frost, as-ymetric, mal-formed, ill-formed, pro-Chomsky
Stemming

• **Normalization Procedure**
  • inflectional morphology:
    • cities ⇒ city, improves/improved ⇒ improve
  • derivational morphology:
    • transformation/transformational ⇒ transform

• **criterion**
  • preserve meaning (word senses)
    • organization ⇔ organ

• **primary application**
  • information retrieval (IR)
  • efficacy questioned: Harman (1991)

Google didn’t use stemming
Stemming and Search

- *up until recently ...*
  - **Word Variations (Stemming)**
    - To provide the most accurate results, Google does not use "stemming" or support "wildcard" searches.
    - In other words, Google searches for exactly the words that you enter in the search box.
    - Searching for "book" or "book*" will not yield "books" or "bookstore". If in doubt, try both forms: "airline" and "airlines," for instance
Stemming and Search

Word Variations (Stemming)

Google now uses stemming technology. Thus, when appropriate, it will search not only for your search terms, but also for words that are similar to some or all of those terms. If you search for "pet lemur dietary needs", Google will also search for "pet lemur diet needs", and other related variations of your terms. Any variants of your terms that were searched for will be highlighted in the snippet of text accompanying each result.

• **Google is more successful than other search engines** in part because it returns “better”, i.e. more relevant, information
  • its algorithm (**a trade secret**) is called **PageRank**

• **SEO (Search Engine Optimization)** is a topic of considerable commercial interest
  • **Goal:** How to get your webpage listed higher by PageRank
  • **Techniques:**
    • e.g. by writing keyword-rich text in your page
    • e.g. by listing morphological variants of keywords

• **Google does not use stemming everywhere**
  • and it does not reveal its algorithm to prevent people “optimizing” their pages
Stemming and Search

• search on:
  • diet requirements
  • (results from 2004)

• note
  • can’t use quotes around phrase
  • blocks stemming
Stemming and Search

• search on:
  • *diet requirements*

• notes
  • Top-ranked page has words *diet*, *dietary* and *requirements*
  • but not the phrase “*diet requirements*”
Stemming and Search

- search on:
  - *diet requirements*

- notes:
  - 5th-ranked page does not have the word *diet* in it
  - an academic conference
    - unlikely to be “optimized” for page hits
  - ranks above 6th-ranked page which does have the exact phrase “*diet requirements*” in it
Stemming and Search

The internet is still growing quickly...

(2008)

Results 1 - 10 of about 636,000 for dietary requirements.

(2011)

About 21,500,000 results (0.14 seconds)

About 95,700,000 results (0.23 seconds)

(2013)

About 29,100,000 results (0.33 seconds)

About 89,200,000 results (0.24 seconds)
Stemming

• **IR-centric view**
  • Applies to open-class lexical items only:
    • *stop-word list*: the, below, being, does
    • exclude: determiners, prepositions, auxiliary verbs

• **not full morphology**
  • prefixes generally excluded
    • *(not meaning preserving)*
    • Examples: asymmetric, undo, encoding
Stemming: Methods

• *use a dictionary* (look-up)
  • OK for English, not for languages with more productive morphology, e.g. Japanese, Turkish, Hungarian etc.

• *write ad-hoc rules*, e.g. Porter Algorithm (Porter, 1980)
  • **Example:**
    • Ends in doubled consonant (not $l$, $s$ or $z$), remove last character
      • *hopping* $\Rightarrow$ *hop*
      • *hissing* $\Rightarrow$ *hiss*
Stemming: Methods

• *dictionary approach not enough*
  • **Example**: (Porter, 1991)
    • routed $\Rightarrow$ route/rout
      • At Waterloo, Napoleon’s forces were *routed*
      • The cars were *routed* off the highway

• **notes**
  • here, the (inflected) verb form is ambiguous
  • preceding word (context) does not disambiguate,
    • *i.e. bigram not sufficient*
Stemming: Errors

- **Understemming**: failure to merge
  - **Example**: adhere/adhesion

- **Overstemming**: incorrect merge
  - **Example**: probe/probable
    - Claim: *-able* irregular suffix, root: *probare* (Lat.)

- **Mis-stemming**: removing a non-suffix (Porter, 1991)
  - **Example**: reply ⇒ rep
Stemming: Interaction

• *interacts with noun compounding*
  • **Example:**
    • operating systems
    • negative polarity items

• for IR, compounds need to be identified first...
Stemming: Porter Algorithm

- The Porter Stemmer (Porter, 1980)
- **Textbook**
  - (page 68, Chapter 3 JM)
  - 3.8 Lexicon- Free FSTs: The Porter Stemmer
- **URL**
  - [http://www.tartarus.org/~martin/PorterStemmer/](http://www.tartarus.org/~martin/PorterStemmer/)
  - for English
  - most widely used system (claimed)
  - manually written rules
  - 5 stage approach to extracting roots
  - considers suffixes only
  - may produce non-word roots

BCPL: extinct programming language

The original stemmer was written in BCPL, a language once popular, but now defunct. For the first few years after 1980 it was distributed in its BCPL form, via the medium of punched paper tape. Versions in other languages soon began to appear, and by 1999 it was being widely used, quoted and adapted. Unfortunately there were numerous variations in functionality among these versions, and this web page was set up primarily to ‘put the record straight’ and establish a definitive version for distribution.
## Stemming: Porter Algorithm

<table>
<thead>
<tr>
<th>language</th>
<th>author</th>
<th>affiliation</th>
<th>received</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSL C</td>
<td>me</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANSL C</td>
<td>me</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>me</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perl</td>
<td>Daniel van den Bemt</td>
<td></td>
<td>Oct 1999</td>
<td>slightly faster?</td>
</tr>
<tr>
<td>Python</td>
<td>Wazid Gupta</td>
<td>The Official Web Guide</td>
<td>Jan 2001</td>
<td></td>
</tr>
<tr>
<td>C Sharp</td>
<td>Andre Hazenhoud</td>
<td>University of Paisley, Scotland</td>
<td>Sep 2004</td>
<td></td>
</tr>
<tr>
<td>C Sharp</td>
<td>Leif Axenrod</td>
<td></td>
<td>Nov 2002</td>
<td></td>
</tr>
<tr>
<td>Common Lisp</td>
<td>Steven M. Hafich</td>
<td>Franz Inc</td>
<td>Mar 2002</td>
<td></td>
</tr>
<tr>
<td>Ruby</td>
<td>Roy Fersia</td>
<td><a href="http://www.ruby-lang.org">www.ruby-lang.org</a></td>
<td>Jan 2003</td>
<td>github link</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>National Institute of Standards and Technology</td>
<td>Bryd University</td>
<td>Apr 2003</td>
<td></td>
</tr>
<tr>
<td>Delphi</td>
<td>J. Babin</td>
<td></td>
<td>Apr 2004</td>
<td></td>
</tr>
<tr>
<td>Javascript</td>
<td>'Andargor'</td>
<td><a href="http://www.andargor.com">www.andargor.com</a></td>
<td>Jul 2004</td>
<td>[see links below]</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>Christian Arden</td>
<td>University of Piraeus, Greece</td>
<td>Jan 2005</td>
<td></td>
</tr>
<tr>
<td>Visual Basic</td>
<td>Phillip Brooks</td>
<td>University of Georgia</td>
<td>Oct 2005</td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>Hareem Anon</td>
<td></td>
<td>Nov 2005</td>
<td></td>
</tr>
<tr>
<td>T-SQL</td>
<td>Keith Barber</td>
<td><a href="http://www.agiledevr.com">www.agiledevr.com</a></td>
<td>May 2006</td>
<td></td>
</tr>
<tr>
<td>Matlab</td>
<td>Juan Carlos Lopez</td>
<td>California Pacific Medical Center Research Institute</td>
<td>Sep 2006</td>
<td></td>
</tr>
<tr>
<td>Tcl</td>
<td>Aria Theodetcios</td>
<td>NCSR Demokritos</td>
<td>Nov 2006</td>
<td></td>
</tr>
<tr>
<td>Erlang</td>
<td>Daniel Tamper</td>
<td>Humboldt-Universitat zu Berlin</td>
<td>May 2007</td>
<td></td>
</tr>
<tr>
<td>Erlang</td>
<td>Alan D.</td>
<td>National Institute of Standards and Technology, Gaithersburg, MD USA</td>
<td>Sep 2007</td>
<td></td>
</tr>
<tr>
<td>RHEL</td>
<td>Daic K Bambroff</td>
<td></td>
<td>Apr 2009</td>
<td></td>
</tr>
<tr>
<td>Scala</td>
<td>Ken Pitsanur</td>
<td></td>
<td>May 2009</td>
<td></td>
</tr>
<tr>
<td>SAS</td>
<td>Antoine St-Pierre</td>
<td>Business Researchers, Inc</td>
<td>Apr 2009</td>
<td></td>
</tr>
<tr>
<td>PL/SQL</td>
<td>Mitchell Roodta</td>
<td></td>
<td>May 2010</td>
<td>github link</td>
</tr>
<tr>
<td>Node.js</td>
<td>Ted Pirogo</td>
<td><a href="http://www.joddio.com">www.joddio.com</a></td>
<td>May 2011</td>
<td>github link</td>
</tr>
<tr>
<td>Google Go</td>
<td>Alex Gogopoulos</td>
<td></td>
<td>Oct 2011</td>
<td>github link</td>
</tr>
<tr>
<td>awk</td>
<td>Gregory Orfanistas</td>
<td>Mds.com/xaxleaf</td>
<td>Jul 2012</td>
<td></td>
</tr>
<tr>
<td>Raku</td>
<td>Yuuki Wada</td>
<td></td>
<td>Mar 2013</td>
<td>bitbucket link</td>
</tr>
<tr>
<td>Rust</td>
<td>Do Ha Ninh</td>
<td>Nanyang Technological University</td>
<td>Aug 2013</td>
<td>github link</td>
</tr>
<tr>
<td>Yava</td>
<td>Sergei Haine</td>
<td></td>
<td>Sep 2013</td>
<td></td>
</tr>
</tbody>
</table>
Stemming: Porter Algorithm

bash-3.2$ /opt/local/bin/swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.2.6)
Copyright (c) 1990-2012 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY and you are welcome to redistribute
Please visit http://www.swi-prolog.org

For help, use ?- help(Topic). or ?- help.

?- use_module(library(porter_stem)).
% library(porter_stem) compiled into true.

porter_stem(+In, -Stem)

Determine the stem of In. In must represent ISO Latin-1 text. The porter_stem/2 predicate first
maps In to lower case, then removes all accents as in unaccent_atom/2 and finally applies the
Porter stem algorithm.

unaccent_atom(+In, -ASCII)

If In is general ISO Latin-1 text with accents, ASCII is unified with a plain ASCII version of the
string. Note that the current version only deals with ISO Latin-1 atoms.

tokenize_atom(+In, -TokenList)

Break the text In into words, numbers and punctuation characters. Tokens are created to the following rules:

{[-]+[0-9]+{|[\-]+}|[0-9]++?([eE][+-]|)[0-9]+?} number
{[\alph{a-zA-Z}]+|[\d][\alpha-zA-Z]+} word
{[\s]}+ skipped
anything else single-character

Character classification is based on the C-library isalnum() etc. functions.
It is likely that future versions of this library will provide tokenize_atom/3 with additional
options to modify space handling as well as the definition of words.

atom_to_stem_list(+In, -ListOfStems)
Stemming: Porter Algorithm

Test it on...

- **Stuff it should work on... regular morphology**
  - agreed, agrees, agreeing, feed
  - care, cares, cared, caring
  - cancellation, cancelation, canceled, cancelled
  - wares, spyware

- **New words**
  - obamacare, cancelization, phablet(s)

- **Understemming**
  - agreement \(\text{(VC)}^1\text{VCC}\) - step 4 (m>1)
  - adhesion
  - geese

- **Overstemming**
  - relativity - step 2

- **Mis-stemming**
  - wander \(\text{C(VC)}^1\text{VC}\)
Porter Stemmer: Perl

```perl
#!/usr/bin/perl -w

# Porter stemmer in Perl. Few comments, but it's easy to follow against the rules in the original
# paper, in
#
# Porter, 1980, An algorithm for suffix stripping, Program, Vol. 14,
# no. 3, pp 130–137,
#
# see also http://www.tartarus.org/~martin/PorterStemmer

# Release 1
local %step1list;
llocal %step2list;
llocal ($c, $v, $C, $V, $mrg0, $mrg1, $mrg2, $v);

sub stem
  { my ($stem, $suffix, $firstch);
    my $w = shift;
    if (length($w) < 3) { return $w; } # length at least 3
    # new map initial y to y so that the patterns never treat it as vowel:
    $w =~ /\./; $firstch = $4;
    if ($firstch =~ /y/) { $w = ufirst $w; }

    # Step 1a
    if ($w =~ /x(s|less$)/) { $w=$$.}$1; }
    elsif ($w =~ /([^s])es$/) { $w=$$.}$1; }
    # Step 1b
    if ($w =~ /ends$/) { if ($s =~ /$mrg0/o) { chop($w); } }
    elsif ($w =~ /ed|ing$/)

    # Step 2a
    if ($w =~ /ys$/) { $stem = $'; if ($stem =~ /$v/0) { $w = $stem."i"; }
```

```
> dhcpc-10-142-152-177:ling538-14 sandiway$ perl porter.perl <<EOF
> adhesion
> EOF
> adhes
```

http://tartarus.org/~martin/PorterStemmer/perl.txt
Porter Stemmer: Perl

```perl
# Step 2
if ($w =~ /(a|e|i|o|u)l|e(w|y)l|e(i|l|y)z|e(st|or)|a|is|ness|nes|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|ness|n
```
Porter Stemmer: Perl

sub initialise {
    # step2list =
    ( 'ational' => 'ate', 'tional' => 'tion', 'enci' => 'ence', 'anci' => 'ance', 'izer' => 'ize', 'bli' => 'ble',
      'alli' => 'al', 'entli' => 'ent', 'oli' => 'o', 'ousli' => 'ous', 'ization' => 'ize', 'ation' => 'ate',
      'ator' => 'ate', 'alize' => 'ai', 'iveness' => 'ive', 'fulness' => 'ful', 'ousness' => 'ous', 'cliti' => 'al',
      'iviti' => 'ive', 'biliti' => 'ble', 'logi' => 'log');
    # step3list =
    ( 'icate' => 'ic', 'ative' => 'a', 'alize' => 'ai', 'iciti' => 'ic', 'ical' => 'ic', 'ful' => 'f', 'ness' => '');

    $c = 
      "[aeiou]\";       # consonant
    $v = 
      "[aeiouy]\";     # vowel
    $cc = 
      "$c\"aeiouy\"\";  # consonant sequence
    $vV = 
      "$v\"aeiouy\"\";  # vowel sequence
    $mg0 = 
      "(S(C))?(V)\$C\";  # VC... is m=0
    $mg1 = 
      "(S(C))?(V)\$C(V)\$C\"; # VC(V) is m=1
    $mg2 = 
      "(S(C))?(V)\$C(V)\$C(V)\$C\"; # VCVC... is m>1
    $vV = 
      "(S(C))?(V)\";   # vowel in stem

    # that's the definition. Run initialise() to set things up, then stem($word) to stem $word, as here:

    initialise();
    while (<>)
    {
        /([^a-zA-Z].*/i) ;
        print $1;
        $w = $2;
        unless (/^[a-zA-Z].*/i) { last; }
        $word = lc $1; # turn to lower case before calling:
        $w = $2;
        $word = stem($word);
        print $word;
        redo;
    }
    print "\n";
}
Stemming: Porter Algorithm

• **rule format:**
  - \((condition\ on\ stem)\ \text{suffix}_1 \rightarrow \text{suffix}_2\)
    - *In case of conflict, prefer longest suffix match*

• “Measure” of a word is \(m\) in:
  - \((C)\ (VC)^m\ (V)\)
    - \(C\) = sequence of one or more consonants
    - \(V\) = sequence of one or more vowels

• **examples:**
  - *tree* \quad C(VC)^0V \quad (tr)()(ee)
  - *troubles* \quad C(VC)^2 \quad (tr)(ou)(bl)(e)(s)

Question: is this a regular expression?
Stemming: Porter Algorithm

• Step 1a: *remove plural suffixation*
  • SSES $\Rightarrow$ SS (caresses)
  • IES $\Rightarrow$ I (ponies)
  • SS $\Rightarrow$ SS (caresss)
  • S $\Rightarrow$ (cats)

• Step 1b: *remove verbal inflection*
  • (m>0) EED $\Rightarrow$ EE (agreed, feed)
  • (*v*) ED $\Rightarrow$ (plastered, bled)
  • (*v*) ING $\Rightarrow$ (motoring, sing)
Stemming: Porter Algorithm

• Step 1b: (*contd. for -ed and -ing rules*)
  • AT ⇒ ATE (conflated)
  • BL ⇒ BLE (troubled)
  • IZ ⇒ IZE (sized)

• (*doubled c & ¬(*L v *S v *Z)) ⇒ single c (hopping, hissing, falling, fizzing)
• (m=1 & *cvc) ⇒ E (filing, failing, slowing)

• Step 1c: Y and l
  • (*v*) Y ⇒ l (happy, sky)
Stemming: Porter Algorithm

• **Step 2: Peel one suffix off for multiple suffixes**
  • 
  • (m>0) ATIONAL ⇒ ATE (relational)
  • (m>0) TIONAL ⇒ TION (conditional, rational)
  • (m>0) ENCI ⇒ ENCE (valenci)
  • (m>0) ANCI ⇒ ANCE (hesitanci)
  • (m>0) IZER ⇒ IZE (digitizer)
  • (m>0) ABLI ⇒ ABLE (conformabli) - *able* (step 4)
  • ...
  • (m>0) IZATION ⇒ IZE (vietnamization)
  • (m>0) ATION ⇒ ATE (predication)
  • ...
  • (m>0) IVITI ⇒ IVE (sensitiviti)
Stemming: Porter Algorithm

• Step 3
  • \((m>0)\) ICATE \(\Rightarrow\) IC (triplicate)
  • \((m>0)\) ATIVE \(\Rightarrow\) (formative)
  • \((m>0)\) ALIZE \(\Rightarrow\) AL (formalize)
  • \((m>0)\) ICITI \(\Rightarrow\) IC (electriciti)
  • \((m>0)\) ICAL \(\Rightarrow\) IC (electrical, chemical)
  • \((m>0)\) FUL \(\Rightarrow\) (hopeful)
  • \((m>0)\) NESS \(\Rightarrow\) (goodness)
Stemming: Porter Algorithm

• **Step 4: Delete last suffix**
  - (m>1) AL ⇒ (revival) - revive, see step 5
  - (m>1) ANCE ⇒ (allowance, dance)
  - (m>1) ENCE ⇒ (inference, fence)
  - (m>1) ER ⇒ (airliner, employer)
  - (m>1) IC ⇒ (gyroscopic, electric)
  - (m>1) ABLE ⇒ (adjustable, mov(e)able)
  - (m>1) IBLE ⇒ (defensible, bible)
  - (m>1) ANT ⇒ (irritant, ant)
  - (m>1) EMENT ⇒ (replacement)
  - (m>1) MENT ⇒ (adjustment)
  - ...

Stemming: Porter Algorithm

• **Step 5a: remove e**
  - (m>1) E $\rightarrow$ (probate, rate)
  - (m>1 & ¬*cvc) E $\rightarrow$ (cease)

• **Step 5b: ll reduction**
  - (m>1 & *LL) $\rightarrow$ L (controller, roll)
Stemming: Porter Algorithm

- **Misses** (understemming)
  - Unaffected:
    - *agreement* (VC)\(^1\)VCC - step 4 \((m>1)\)
    - *adhesion*
  - Irregular morphology:
    - drove, geese

- **Overstemming**
  - *relativity* - step 2

- **Mis-stemming**
  - *wander* C(VC)\(^1\)VC

SWI-Prolog/PERL/Python NLTK

- adhes vs. adher(e)
- rel
- wander
Stemming: Porter Algorithm

Extensions?

• *ad hoc* and English-specific

• errors can be fixed on a case-by-case basis
  • i.e. use a look-up table (dictionary)
  • also there are a finite number of irregularities,
    e.g. *drive/drove*, *goose/geese*