Faculty of Computer Science, Dalhousie University *21-Sep-2023*

CSCI 4152/6509 — Natural Language Processing

Lecture 6: Elements of Morphology

Location: Rowe 1011 Instructor: Vlado Keselj
Time: 16:05 – 17:25

Previous Lecture

- Regular expressions in Perl
  - Use of special variables
  - Backreferences, shortest match
- Text processing examples
  - tokenization
  - counting letters

We will look now at an implementation where letters and their frequencies are sorted by the frequency, from the highest-frequency letter to the lowest. We will also produce frequencies both as letter counts, and as normalized frequencies; i.e., as proportional frequencies of the letters out of 1.

Letter Frequencies Modification (3)

```
#!/usr/bin/perl
# Letter frequencies (3)

while (<>) {
  while (/[a-zA-Z]/) {
    my $l = $&; $_ = $';
    $f{lc $l} += 1; $tot ++;
  }
}

for (sort { $f{$b} <=> $f{$a} } keys %f) {
  print sprintf("%6d %.4lf %s\n",
              $f{$_}, $f{$_}/$tot, $_); }
```

Output 3

```
 35697 0.1204 e
 28897 0.0974 t
 23528 0.0793 a
 23264 0.0784 o
 20200 0.0681 n
 19608 0.0661 h
 18849 0.0635 i
 17760 0.0599 s
 15297 0.0516 r
```

```
 14879 0.0502 d
 12163 0.0410 l
  8959 0.0302 u
...
```

6 Elements of Morphology

- Reading: Section 3.1 in the textbook, "Survey of (Mostly) English Morphology"
- *morphemes* — smallest meaning-bearing units
- *stems* and *affixes*; stems provide the "main" meaning, while affixes act as modifiers
- affixes: prefix, suffix, infix, or circumfix
- cliticization — clitics appear as parts of a word, but syntactically they act as words (e.g., 'm, 're, 's)
- tokenization, stemming (Porter stemmer), lemmatization

The *morphemes* are the smallest meaning-bearing parts of a word. For example, the word *cats* contains two morphemes *cat* and *s*, the word *unbelievably* contains the four morphemes *un*, *believ*, *ab*, and *ly*, and the word *unmorpholinguistically* contains the six morphemes *un*, *morpho*, *ling*, *uist*, *ical*, and *ly*. It could be sometimes debatable what is the proper way of breaking a word into morphemes, but not having a clear correct answer is not uncommon in analysis of natural languages.

- suffix example: eat*s*; prefix example: *un*buckle; circumfix example from German: sagen (to say) and *ge*esag*t* (said, past participle); infix example from Tagalog (Philipine language): hingi (borrow) and h*um*ingi
- stacking multiple affixes is possible: unbelievably = un-believe-able-y
- English typically allows up to 4 affixes, but some languages allow up to 10 affixes, such as Turkish. Such languages are are called *agglutinative* languages.
- *cliticization* is considered to be a morphological process
- Clitics appear as orthographic or phonological parts of the words, but syntactically they act as words.
- Clitic examples: 'm in I'm, 're in we're, possessive 's

Tokenization

- Text processing in which plain text is broken into words or *tokens*
- Tokens include non-word units, such as numbers and punctuation
- Tokenization may normalize words by making them lower-case or similar
- Usually simple, but prone to ambiguities, as most of the other NLP tasks

Tokenization is text processing in which the plain text is broken into words. It may not be a simple process, depending on the type of text and kind of tokens that we want to recognize.

Stemming is the type of word processing in which a word is mapped into its *stem*, which is a part of the word that represents the main meaning of the word. For example, *foxes* is mapped to the stem *fox*, or the word *semantically* is mapped to the stem *semanti*.

It is used in Information Retrieval due to the property that if two words have the same stem, they are typically semantically very related. Hence, if words in documents and queries are replaced by their stems, the resulting indices are smaller, and words in a query can be easily matched with their morphological variations.

Lemmatization is a word processing method in which a *surface word form*, i.e., the word form as it appears in text, is mapped to its *lemma*, i.e., the canonical form as it appears in a dictionary. For example, the word *working* would be mapped into the verb *work*, or the word *semantically* would be mapped to the lemma *semantics*.

6.1 Morphological Processes

A *morphological process* is a word transformation that happens as a regular language transformation. There are tree main morphological processes in English:

1. inflection,
2. derivation, and
3. compounding.

1. Inflection: is a transformation that transforms a word from one lexical class into another related word in the same class. The transformation is performed by adding or changing a suffix or prefix. It is highly regular transformation. Some inflection examples are: dog → dogs, work → works, work → working, and work → worked.

We will discuss more the concept of *lexical class* or *part of speech* class later, but for now you are probably familiar with the following lexical classes (or types of words): nouns, verbs, adjectives, adverbs, and maybe some other.

Inflection is so regular transformation that usually we do not find inflected variations of a word in a dictionary. It is assumed that a reader of the dictionary will be able to derive these variations by herself. Similarly, we can frequently program inflection in a computer application rather than storing different variations of the word.

2. Derivation: is a transformation that transforms a word from one lexical class into a related word in a different class. Similarly to inflection, it is performed by adding or changing a suffix or prefix. There is also some regularity, but it is less regular than inflection. For example, a derivation is *wide (adjective) → widely (adverb)*, but a similar transformation *old → oldly* is not valid. Some other examples are: accept (verb) → acceptable (adjective), acceptable (adjective) → acceptably (adverb), and teach (verb) → teacher (noun).

There are exceptions where a derivation is used to transform a word in a lexical class to another word in the same class but it is a significantly a different word. For example, the transformation of the adjective *red* to *redish* is considered a derivation, rather than an inflection.

Since derivation is not as regular transformation as inflection, derived variations of a word are usually stored in a dictionary, and in a computer application we may want to store them in a lexicon, i.e., a word database, in many cases.

Below you can find a table with some more derivation examples:

Derivation type	Suffix	Example		
noun-to-verb	*-fy*	glory	→	glorify
noun-to-adjective	*-al*	tide	→	tidal
verb-to-noun (agent)	*-er*	teach	→	teacher
verb-to-noun (abstract)	*-ance*	delivery	→	deliverance
verb-to-adjective	*-able*	accept	→	acceptable
adjective-to-noun	*-ness*	slow	→	slowness
adjective-to-verb	*-ise*	modern	→	modernise (Brit.)
adjective-to-verb	*-ize*	modern	→	modernize (U.S.)
adjective-to-adjective	*-ish*	red	→	reddish
adjective-to-adverb	*-ly*	wide	→	widely

3. Compounding: is a transformation where two or more words are combined, usually by concatenation, to create a new word. Some examples are: news + group → newsgroup, down + market → downmarket, over + take → overtake, play + ground → playground, and lady + bug → ladybug.

7 Characters, Words, and N-grams

7.1 Zipf's Law

- We looked at code for counting letters, words, and sentences
- We can look again at counting words; e.g., in "Tom Sawyer":
- We can observe: Zipf's law (1929): $r \times f \approx$ const.

Word	Freq (f)	Rank (r)
the	3331	1
and	2971	2
a	1776	3
to	1725	4
of	1440	5
was	1161	6
it	1030	7
I	1016	8
that	959	9
he	924	10
in	906	11
's	834	12
you	780	13
his	772	14
Tom	763	15
't	654	16
⋮	⋮	

One of the basic tasks that we can do using stream-oriented processing of language is to collect statistical values on letters, words, sentences, or similar tokens. We saw previously the code for finding frequency of different letters, and these data can be useful for example for computer identification of a natural language. We can do similar counting but this time of word frequencies. The table above shows the frequencies of words in the novel "Tom Sawyer" by Mark Twain.

Zipf's law is an observation that the product of rank and frequency of the words in a text is "quite constant," if we can use that term. For example, we can test this "law" on the words in the "Tom Sawyer" novel using the following code:

Counting Words

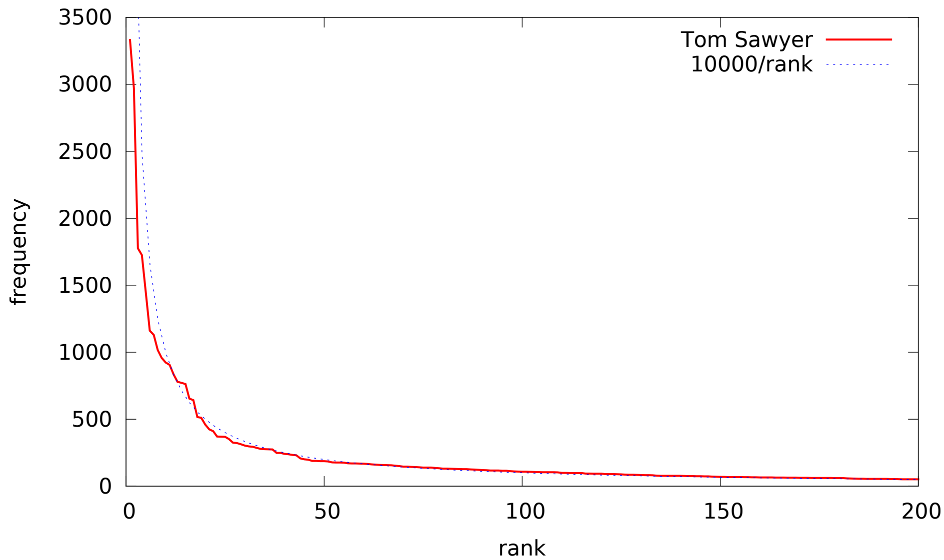
```
#!/usr/bin/perl
# word-frequency.pl

while (<>) {
  while (/'?[a-zA-Z]+/g) { $f{$&}++; $tot++; }
}

print "rank   f   f(norm) word       r*f\n".
      ('-'x35)."\n";
for (sort { $f{$b} <=> $f{$a} } keys %f) {
  print sprintf("%3d. %4d %lf %-8s %5d\n",
              ++$rank, $f{$_}, $f{$_}/$tot, $_,
              $rank*$f{$_});
}
```

Program Output (Zipf's Law)

```
rank   f    word       r*f       18.  516   for       9288
```

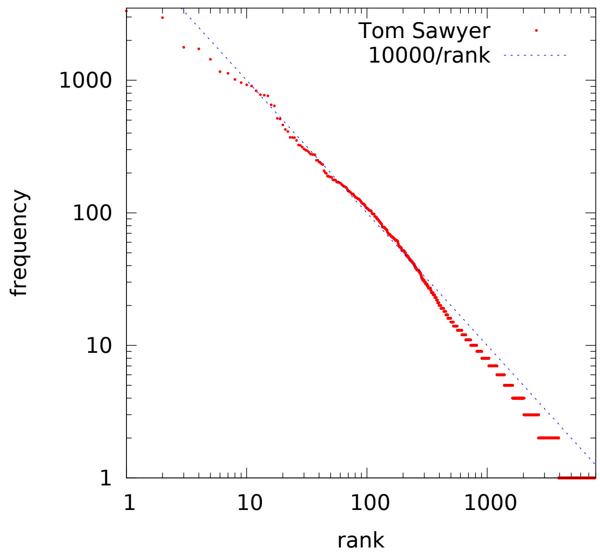
1.	3331	the	3331
2.	2971	and	5942
3.	1776	a	5328
4.	1725	to	6900
5.	1440	of	7200
6.	1161	was	6966
7.	1130	it	7910
8.	1016	I	8128
9.	959	that	8631
10.	924	he	9240
11.	906	in	9966
12.	834	's	10008
13.	780	you	10140
14.	772	his	10808
15.	763	Tom	11445
16.	654	't	10464
17.	642	with	10914
19.	511	had	9709
20.	460	they	9200
21.	425	him	8925
22.	411	but	9042
23.	371	on	8533
24.	370	The	8880
25.	369	as	9225
26.	352	said	9152
27.	325	He	8775
28.	322	at	9016
29.	313	she	9077
30.	303	up	9090
31.	297	so	9207
32.	294	be	9408
33.	286	all	9438
34.	278	her	9452
35.	276	out	9660
36.	275	not	9900

We can present this data in a graphical form and compare it with the function $f = 10000/r$ to demonstrate the



Zipf's law:

If we apply a logarithm on both sides of the Zipf's formula we get the formula $\log r + \log f \approx$ const., which means that the Zipf's law implies that the rank-frequency graph using log scales of x and y axis should be close to a straight line, descending under an angle of 45 degrees. The following graph illustrates this:


Tom Sawyer
10000/rank
frequency
rank
1
10
100
1000
1
10
100
1000