STEMMING AND LEMMATIZATION

CS 562/662: Natural Language Processing

2015-01-20

Stemming and *lemmatization* address the intuition that much of inflectional morphology is irrelevant for information retrieval and models of word similarity. E.g.: someone who searches for *pizzas* is probably just as interested in pages that only mention *pizza*.

GLOSSARY

Wordform: the surface form of a word

Paradigm: a set of morphologically related wordforms

Lemma: the “citation form” of a paradigm, itself a wordform

LEMMATIZATION

Lemmatization is a procedure in which wordforms are replaced with their lemma, usually via a fixed lookup table. (Out-of-vocabulary words are usually left as is.)

STEMMING

Stemming is a procedure which attempts to strip inflectional (and, sometimes, derivational) affixes from orthographic tokens, producing *stems*. These need not be identical to the lemma or any other form of the paradigm, but wordforms which share a stem are very likely morphologically related.

RULES FOR THE PORTER ENGLISH STEMMER

- Step 1a:

```
SSES# →  SS#    (caresses  →  caress)
 IES# →  I#     (ponies    →  poni)
  SS# →  SS#    (caress    →  caress)
   S# →  _      (cats      →  cat)
```

…

- Step 5a:

```
E# →  __          (awake     →  awak)
```

if the stem contains two or more VC sequences or the stem ends in C_1VC_2 where C_2 is not W, X, or Y

[Source: Porter 1980]

THE PORTER STEMMER IN ACTION

	controller	playfulness	indefatigability
Step 1			indefatigabiliti
Step 2		playful	indefatigable
Step 3		play	
Step 4	controll		indefatig
Step 5	control		

MULTILINGUAL SUPPORT

- The Snowball stemmer collection (Porter 2001) supports:

  Armenian, Basque, Catalan, Czech, Danish, Dutch, English, Finnish, French, German, Hungarian, Irish, Italian, Latin (!), Norwegian, Portuguese, Romanian, Russian, Spanish, Swedish, and Turkish (!)

- Stemmers written with the Snowball DSL can be compiled down to C and Java (a.o.) functions

[Source: snowball.tartarus.org]

PYTHON STEMMING API

```
>>> from nltk.stem.snowball import
SnowballStemmer
>>> STEM = SnowballStemmer("english").stem
>>> tokens = "I tell you that there are
terrible temptations which it requires
strength and courage to yield to".split()
>>> stems = [STEM(token) for token in tokens]
>>> print " ".join(stems).upper()
I TELL YOU THAT THERE ARE TERRIBL TEMPTAT
WHICH IT REQUIR STRENGTH AND COURAG TO YIELD
TO
```

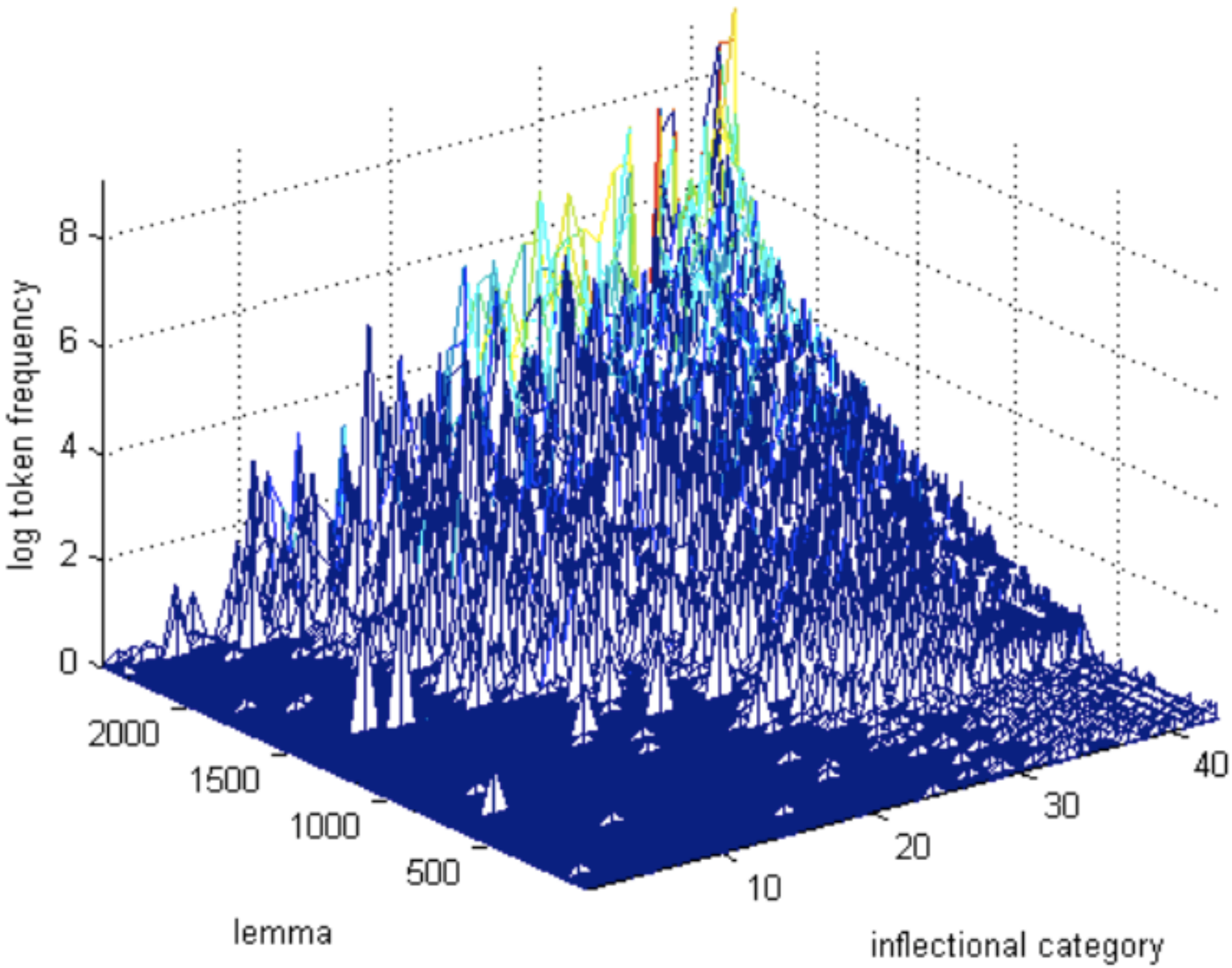
One weakness of stemming (compared to lemmatization) is that it does not attempt to handle irregular morphology (e.g., *sing-sung*). To leverage the strengths of both techniques, first apply lemmatization, then stemming.

These methods work well in English simply because it has less morphology than just about any other human language—it gets worse from here.

ZIPF'S LAW IN EUROPEAN VERB PARADIGMS

	Tokens (Mil)	Unique Vfs	Max Vfs per	Saturation %
English	1.3	6	6	100.0
Italian	1.4	55	47	85.5
Slovene	2.4	32	24	75.0
Catalan	1.7	45	33	73.3
Basque	0.6	22	16	72.7
Hebrew	2.5	33	23	69.7
Spanish	2.6	51	34	66.7
Swedish	1.0	21	14	66.7
Hungarian	1.2	76	48	63.2
Czech	2.0	72	41	56.9
Greek	2.8	83	45	54.2
Finnish	2.1	365	147	40.3

[Source: Chan 2008]



[Source: Chan 2008 (Spanish)]

It is also possible to perform full-scale morphological analysis to isolate stems, but it this is not often done in practice: it is considerably slower, may require labeled training data, and is not particularly precise.