Lesson 12: Processing a Corpus

Fundamentals of Text Processing for Linguists

Na-Rae Han

Objectives

- How to process an ARCHIVE of text, i.e., corpus?
- Fun with CMU Pronouncing Dictionary

Processing multiple texts

- From the NLTK Corpora page, download:
  - *C-Span Inaugural Address Corpus*
  - http://nltk.googlecode.com/svn/trunk/nltk_data/index.xml

- **The C-Span Inaugural Address Corpus**
  - Includes 56 past presidential inaugural address, from 1789 (Washington) to 2009 (Obama).
  - The directory has **56 .txt files** and **one README file**.

- QUESTION: How do we effectively process this many files?

Using glob

- glob: a file-name globbing utility
  - Returns **a list of file names** that match the specified pattern

```
>>> import glob
>>> fnames = glob.glob(r'D:\Lab\inaugural\*.txt')
>>> len(fnames)
56
>>> fnames[:5]
['D:\\Lab\\inaugural\\1789-Washington.txt',
'D:\\Lab\\inaugural\\1793-Washington.txt',
'D:\\Lab\\inaugural\\1797-Adams.txt',
'D:\\Lab\\inaugural\\1801-Jefferson.txt',
'D:\\Lab\\inaugural\\1805-Jefferson.txt']
>>> fnames[-1]
'D:\\Lab\\inaugural\\2009-Obama.txt'
>>>
```

Using glob

- Addresses from 1800's only

```
>>> fnames2 = glob.glob(r'D:\Lab\inaugural\18*.txt')
>>> len(fnames2)
25
>>> fnames2[:5]
['D:\\Lab\\inaugural\\1801-Jefferson.txt',
'D:\\Lab\\inaugural\\1805-Jefferson.txt',
'D:\\Lab\\inaugural\\1809-Madison.txt',
'D:\\Lab\\inaugural\\1813-Madison.txt',
'D:\\Lab\\inaugural\\1817-Monroe.txt']
>>> fnames2[-1]
'D:\\Lab\\inaugural\\1897-McKinley.txt'
>>>
```

Treating files as a single corpus

- Task: Compile **word frequency** of the Inaugural Speeches.
  - For this, we only need to build a single pool of tokenized words.
  - For each file name, open it, read in the text, tokenize the text, and then add the result to the pool of tokenized words.

```
>>> import textproc
>>> alltoks = []
>>> for fn in fnames:
        f = open(fn)
        ftxt = f.read()
        f.close()
        ftoks = textproc.getToks(ftxt)
        alltoks.extend(ftoks)

>>> len(alltoks)
145774
>>>
```

Word frequency of entire corpus

```
>>> alltoks[:15]
['fellow', '-', 'citizens', 'of', 'the', 'senate', 'and', 'of',
'the', 'house', 'of', 'representatives', ':', 'among', 'the']
>>> alltoks[-15:]
['you', '.', 'god', 'bless', 'you', '.', 'and', 'god', 'bless',
'the', 'united', 'states', 'of', 'america', '.']
>>> allfreq = textproc.getFreq(alltoks)
>>> for k in sorted(allfreq, key=allfreq.get, reverse=True)[:10]:
        print k, allfreq[k]

the 9906
of 6986
, 6862
and 5139
. 4749
to 4432
in 2749
a 2193
our 2058
that 1726
```

Processing each text file

- Task: Compute the **average sentence length** for **each presidential address**.
  - ⬅ We have to build separate token lists for each file.

```
>>> fn2toks = {}
>>> for fn in fnames:
        f = open(fn)
        ftxt = f.read()
        f.close()
        ftoks = textproc.getToks(ftxt)
        fn2toks[fn] = ftoks

>>> len(fn2toks)
56
>>> fnames[0]
'D:\\Lab\\inaugural\\1789-Washington.txt'
>>> fn2toks['D:\\Lab\\inaugural\\1789-Washington.txt']
['fellow', '-', 'citizens', 'of', 'the', 'senate', 'and', 'of',
'the', 'house', 'of', 'representatives', ':', ...
```

Average sentence length, per address

```
>>> for fn in sorted(fn2toks):
        toks = fn2toks[fn]
        sentcount = toks.count('.') + toks.count('!') \
                    + toks.count('?')
        avgsentlen = len(toks)/float(sentcount)
        print avgsentlen, '\t', fn[fn.rindex('\\')+1:]


66.9130434783  1789-Washington.txt
36.75  1793-Washington.txt
69.8648648649  1797-Adams.txt
47.1951219512  1801-Jefferson.txt
52.9777777778  1805-Jefferson.txt
60.2380952381  1809-Madison.txt
...
18.824742268   2001-Bush.txt
23.3939393939  2005-Bush.txt
24.7909090909  2009-Obama.txt
>>>
```

Fun with CMU Pronouncing Dictionary

- Load your saved pickle file as cmudict.
  - If you didn't pickle yours, download:
    http://www.pitt.edu/~naraehan/ling1991/cmudict.zip

```
>>> cmudict['HELLO']
[['HH', 'AH0', 'L', 'OW1'],
['HH', 'EH0', 'L', 'OW1']]
>>>
```

- cmudict challenges:
  - Find all words that have both /θ/ and /ð/.
  - Find all words that have both /ʃ/ and /ʒ/.
  - Find all words, like 'anxious', ending in /kʃəs/.
  - Find all words that rhyme with 'orange'.

```
>>> for w in cmudict:
        prons = cmudict[w]
        for p in prons:
            if 'TH' in p and 'DH' in p:
                print w, p
```

Words with /θ/ and /ð/

```
THENCEFORTH ['DH', 'EH2', 'N', 'S', 'F', 'AO1', 'R', 'TH']
>>>
>>> for w in cmudict:
        prons = cmudict[w]
        for p in prons:
            if 'SH' in p and 'ZH' in p:
                print w, p
```

Words with /ʃ/ and /ʒ/

```
VISUALIZATION ['V', 'IH2', 'ZH', 'W', 'AH0', 'L', 'AH0', 'Z', 'EY1', 'SH', 'AH0', 'N']
SCHLUMBERGER ['SH', 'L', 'AH1', 'M', 'B', 'ER0', 'ZH', 'EY2']
JEANMICHELE ['ZH', 'AA2', 'N', 'M', 'AH0', 'SH', 'EH1', 'L']
SHENZHEN ['SH', 'EH1', 'N', 'ZH', 'EH2', 'N']
>>>
```

```
>>> cmudict['ANXIOUS']
[['AE1', 'NG', 'K', 'SH', 'AH0', 'S'], ['AE1', 'NG', 'SH', 'AH0', 'S']]
>>> for w in cmudict:
        prons = cmudict[w]
        for p in prons:
                if p[-4:] == ['K', 'SH', 'AH0', 'S']:
                        print w, p
```

Words ending in /kʃəs/

```
NOXIOUS ['N', 'AA1', 'K', 'SH', 'AH0', 'S']
OBNOXIOUS ['AA0', 'B', 'N', 'AA1', 'K', 'SH', 'AH0', 'S']
INFECTIOUS ['IH0', 'N', 'F', 'EH1', 'K', 'SH', 'AH0', 'S']
FRACTIOUS ['F', 'R', 'AE1', 'K', 'SH', 'AH0', 'S']
ANXIOUS ['AE1', 'NG', 'K', 'SH', 'AH0', 'S']
RAMBUNCTIOUS ['R', 'AE0', 'M', 'B', 'AH1', 'NG', 'K', 'SH', 'AH0', 'S']
>>>
```

```
>>> cmudict['ORANGE']
[['AO1', 'R', 'AH0', 'N', 'JH'], ['AO1', 'R', 'IH0', 'N', 'JH']]
>>> for w in cmudict:
        prons = cmudict[w]
        for p in prons:
                if p[-3:] == ['AH0', 'N', 'JH']:
                        print w, p

COUNTERCHALLENGE ['K', 'AW1', 'N', 'T', 'ER0', 'CH', 'AE2', 'L', 'AH0', 'N', 'JH']
COUNTERCHALLENGE ['K', 'AW1', 'N', 'ER0', 'CH', 'AE2', 'L', 'AH0', 'N', 'JH']
LOZENGE ['L', 'AO1', 'Z', 'AH0', 'N', 'JH']
CHALLENGE ['CH', 'AE1', 'L', 'AH0', 'N', 'JH']
ALONGE ['AE1', 'L', 'AH0', 'N', 'JH']
SCAVENGE ['S', 'K', 'AE1', 'V', 'AH0', 'N', 'JH']
ORANGE ['AO1', 'R', 'AH0', 'N', 'JH']
>>>
```

Rhymes with *orange,* /ə n dʒ/
/ə / is unstressed: AH**0**

```
>>> cmudict['ORANGE']
[['AO1', 'R', 'AH0', 'N', 'JH'], ['AO1', 'R', 'IH0', 'N', 'JH']]
>>> for w in cmudict:
        prons = cmudict[w]
        for p in prons:
                if p[-3].startswith('IH') and p[-2:] == ['N', 'JH'] :
                        print w, p

['K', 'AA1', 'L', 'IH0', 'N', 'JH'] collinge
['AO1', 'R', 'IH0', 'N', 'JH'] orange
['M', 'IH1', 'N', 'JH'] minge
['F', 'R', 'IH1', 'N', 'JH'] fringe
['AH0', 'N', 'HH', 'IH1', 'N', 'JH'] unhinge
['HH', 'IH1', 'N', 'JH'] hinge
['S', 'IH1', 'N', 'JH'] singe
['B', 'IH1', 'N', 'JH'] binge
['K', 'L', 'IH1', 'N', 'JH'] klinge
['IH0', 'M', 'P', 'IH1', 'N', 'JH'] impinge
['K', 'R', 'IH1', 'N', 'JH'] cringe
['IH1', 'N', 'JH'] inge
['V', 'IH1', 'N', 'JH'] vinje
['S', 'ER0', 'IH1', 'N', 'JH'] syringe
['S', 'IH1', 'R', 'IH0', 'N', 'JH'] syringe
['W', 'IH1', 'N', 'JH'] winge
['IH0', 'N', 'F', 'R', 'IH1', 'N', 'JH'] infringe
['T', 'W', 'IH1', 'N', 'JH'] twinge
['T', 'IH1', 'N', 'JH'] tinge
```

Rhymes with *orange*, /ɪ n dʒ/, STRESS is ignored

HW#4: Two ESL Corpora

Bulgarian Students

It is time, that our society is dominated by industrialization. The prosperity of a country is based on its enormous industrial corporations that are gradually replacing men with machines. Science is highly developed and controls the economy. From the beginning of school life students are expected to master a huge amount of scientific data. Technology is part of our everyday life.

Children nowadays prefer to play with computers rather than with our parents' wooden toys. But I think that in our modern world which worships science and technology there is still a place for dreams and imagination.

There has always been a place for them in man's life. Even in the darkness of the …

Japanese Students

I agree greatly this topic mainly because I think that English becomes an official language in the not too distant. Now, many people can speak English or study it all over the world, and so more people will be able to speak English. Before the Japanese fall behind other people, we should be able to speak English, therefore, we must study English not only junior high school students or over but also pupils. Japanese education system is changing such a program. In this way, Japan tries to internationalize rapidly. However, I think this way won't suffice for becoming international humans. To becoming international humans, we should study English not only school but also daily life. If we can do it, we are able to master English conversation. It is important for us to master English honorific words. …

Wrap-up

- **Next class**
  - Unicode handling
  - Object-oriented programming (presented by Shameek)

- **Homework #4**
  - **Bulgarian vs. Japanese ESL essays**