Lesson 10: List Comprehension, Working with Large Texts

Fundamentals of Text Processing for Linguists

Na-Rae Han

Objectives

- Review of Homework #3

- List comprehension
- Practice:
  - working with your custom module in IDLE shell
  - working with large text files

HW#3: Washington vs. Obama

Washington (1789)

Token count: 1539

Type count: 603

TTR: 0.391812865497

Obama (2009)

Token count: 2727

Type count: 899

TTR: 0.329666299963

Washington's TTR is higher. Richer vocabulary in Washington?
➔ NOPE. Washington's speech is MUCH shorter!

Sentence count: 23

Average sent length: 66.91

Average word length (symbols excluded): 4.94

Sentence count: 110

Average sent length: 24.79

Average word length (symbols excluded): 4.41

Top 20 words

Washington (1789)

the	116	0.0753736192333
of	71	0.0461338531514
,	70	0.0454840805718
to	48	0.0311890838207
and	48	0.0311890838207
which	36	0.0233918128655
in	31	0.0201429499675
.	23	0.0149447693307
i	23	0.0149447693307
be	23	0.0149447693307
my	22	0.0142949967511
by	20	0.0129954515919
that	18	0.0116959064327
with	17	0.0110461338532
on	15	0.00974658869396
a	14	0.00909681611436
as	14	0.00909681611436
have	12	0.00779727095517
for	12	0.00779727095517
it	11	0.00714749837557

Obama (2009)

the	135	0.049504950495
,	130	0.04767143381
and	111	0.040704070407
.	109	0.039970663733
of	82	0.030069673634
to	70	0.02566923359
our	67	0.024569123579
we	62	0.022735606894
that	49	0.017968463513
a	47	0.017235056839
is	36	0.013201320132
in	25	0.00916758342501
this	24	0.00880088008801
for	23	0.00843417675101
us	23	0.00843417675101
--	22	0.00806747341401
;	22	0.00806747341401
are	22	0.00806747341401
but	20	0.00733406674001
will	19	0.00696736340301

Only in one speech

Washington (1789)			Obama (2009)		
me	8	0	'	0	13
present	5	0	s	0	12
under	5	0	america	0	10
being	4	0	because	0	8
duty	4	0	what	0	8
myself	4	0	do	0	7
ought	4	0	let	0	7
since	4	0	cannot	0	6
into	3	0	common	0	6
measures	3	0	today	0	6

Top 20 words favored by

Washington (1789)

the	116	135	0.0258686
which	36	4	0.0219249
of	71	82	0.0160641
i	23	3	0.0138446
my	22	2	0.0135615
in	31	25	0.0109753
be	23	12	0.0105443
by	20	8	0.0100618
with	17	13	0.0062789
to	48	70	0.0055198
an	10	3	0.0053976
as	14	11	0.0050630
your	9	3	0.0047478
government	8	3	0.0040980
public	6	1	0.0035319
on	15	17	0.0035126
more	8	5	0.0033646
every	9	8	0.0029143
citizens	5	1	0.0028821
his	5	1	0.0028821

Obama (2009)

.	23	109	-0.0250258
our	1	67	-0.0239193
we	1	62	-0.0220858
and	48	111	-0.0095149
is	7	36	-0.0086529
a	14	47	-0.0081382
us	1	23	-0.0077844
--	1	22	-0.0074177
that	18	49	-0.0062725
are	3	22	-0.0061181
but	3	20	-0.0053847
they	3	17	-0.0042846
new	1	11	-0.0033839
not	4	16	-0.0032681
who	3	14	-0.0031845
nation	2	12	-0.0031008
;	8	22	-0.0028692
-	1	9	-0.0026505
this	10	24	-0.0023031
,	70	130	-0.0021873

Anything else?

- What discoveries did you make?

- What sort of linguistic inquiry is beyond our current Python capability?
  - Who uses more adjectives/adverbs/function words…?

    ← We need a part-of-speech (POS) tagger.
  - Who uses more relative clauses?

    ← We need a syntactic parser.
  - Who likes "rarer" vocabulary rather than plain words?

    ← We need a large-scale English vocabulary ranking

Filtering a list

```
>>> mary = 'Mary had a little lamb, whose fleece was white as
snow.'.split()
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

- How to make a list of words that have 'a'?

```
>>> alist = []
>>> for w in mary:
        if 'a' in w:
                alist.append(w)


>>> alist
['Mary', 'had', 'a', 'lamb,', 'was', 'as']
```

You need to make a new empty list, and then iterate through mary to find items to put in

Filtering a list

```
>>> mary = 'Mary had a little lamb, whose fleece was white as
snow.'.split()
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

- How to make a list of words that have 'a'?

```
>>> [w for w in mary if 'a' in w]
['Mary', 'had', 'a', 'lamb,', 'was', 'as']
>>>
```

The power of
LIST COMPREHENSION

- Creating a new list where elements meet a certain condition:

```
[x for x in list if ... ]
```

Try it out

2 minutes

```
>>> mary = 'Mary had a little lamb, whose fleece was white as
snow.'.split()
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

- Syntax: `[x for x in list if ... ]`

Words that have 'a'

Words that are 5 chars or longer

Words that are 5 chars or longer and without symbols

```
>>> [w for w in mary if 'a' in w]
['Mary', 'had', 'a', 'lamb,', 'was', 'as']

>>> [w for w in mary ??
['little', 'lamb,', 'whose', 'fleece', 'white',
'snow.']

>>> [w for w in mary ?? ]
['little', 'whose', 'fleece', 'white']
```

Try it out

2 minutes

```
>>> mary = 'Mary had a little lamb, whose fleece was white as
snow.'.split()
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

- Syntax: `[x for x in list if ... ]`

Words that have 'a'

```
>>> [w for w in mary if 'a' in w]
['Mary', 'had', 'a', 'lamb,', 'was', 'as']
```

Words that are 5 chars or longer

```
>>> [w for w in mary if len(w) >=5]
['little', 'lamb,', 'whose', 'fleece', 'white',
'snow.']
```

Words that are 5 chars or longer and without symbols

```
>>> [w for w in mary if len(w) >=5 and w.isalnum()]
['little', 'whose', 'fleece', 'white']
```

Transforming items in list

```
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

- How to make a new list with uppercase words?

```
>>> mary.upper()
. . .
AttributeError: 'list' object has no attribute 'upper'
>>> mup = []
>>> for w in mary:
        mup.append(w.upper())

>>> mup
['MARY', 'HAD', 'A', 'LITTLE', 'LAMB,', 'WHOSE',
'FLEECE', 'WAS', 'WHITE', 'AS', 'SNOW.']
```

Cannot uppercase a list

You have to create an empty new list and then put in uppercased words

Transforming items in list

```
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

- Uppercased list, using list comprehension

```
>>> [w.upper() for w in mary]
['MARY', 'HAD', 'A', 'LITTLE', 'LAMB,', 'WHOSE', 'FLEECE', 'WAS',
'WHITE', 'AS', 'SNOW.']
>>>
```

- Creating a new list where each element is transformed:

```
[f(x) for x in list]
```

Try it out

2 minutes

```
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

- Syntax: `[f(x) for x in list]`

List of first characters

List of word lengths

List of True/False for having 'a' as substring

```
>>> [w.upper() for w in mary]
['MARY', 'HAD', 'A', 'LITTLE', 'LAMB,', 'WHOSE',
'FLEECE', 'WAS', 'WHITE', 'AS', 'SNOW.']
>>> [ ? for w in mary]
['M', 'h', 'a', 'l', 'l', 'w', 'f', 'w', 'w',
'a', 's']
>>> [ ? for w in mary]
[4, 3, 1, 6, 5, 5, 6, 3, 5, 2, 5]
>>> [ ? for w in mary]
[True, True, True, False, True, False, False,
True, False, True, False]
```

Try it out

2 minutes

```
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

- Syntax: `[f(x) for x in list]`

List of first characters

List of word lengths

List of True/False for having 'a' as substring

```
>>> [w.upper() for w in mary]
['MARY', 'HAD', 'A', 'LITTLE', 'LAMB,', 'WHOSE',
'FLEECE', 'WAS', 'WHITE', 'AS', 'SNOW.']
>>> [w[0] for w in mary]
['M', 'h', 'a', 'l', 'l', 'w', 'f', 'w', 'w',
'a', 's']
>>> [len(w) for w in mary]
[4, 3, 1, 6, 5, 5, 6, 3, 5, 2, 5]
>>> ['a' in w for w in mary]
[True, True, True, False, True, False, False,
True, False, True, False]
```

Try it out

2 minutes

```
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

Words that are 6 chars or longer, in upper case

```
>>> [ ?? ]
['LITTLE', 'FLEECE']
```

Calculate the average word length … in one line!

```
>>> [len(w) for w in mary]
[4, 3, 1, 6, 5, 5, 6, 3, 5, 2, 5]
>
??
```

Try it out

2 minutes

```
>>> mary
['Mary', 'had', 'a', 'little', 'lamb,', 'whose', 'fleece', 'was',
'white', 'as', 'snow.']
```

Words that are 6 chars or longer, in upper case

```
>>> [w.upper() for w in mary if len(w) >= 6]
['LITTLE', 'FLEECE']
```

Calculate the average word length … in one line!

```
>>> [len(w) for w in mary]
[4, 3, 1, 6, 5, 5, 6, 3, 5, 2, 5]
>>> sum([len(w) for w in mary])
45
>>> sum([len(w) for w in mary]) / len(mary)
4
>>> sum([len(w) for w in mary]) / float(len(mary))
4.090909090909091
```

List comprehension: summary

- Syntax: `[f(x) for x in list if ...]`

```
>>> mary = 'Mary had a little lamb'.split()
>>> mary
['Mary', 'had', 'a', 'little', 'lamb']

>>> [w for w in mary]
['Mary', 'had', 'a', 'little', 'lamb']

>>> [w for w in mary if len(w) > 3]
['Mary', 'little', 'lamb']
>>> [w for w in mary if 'a' in w]
['Mary', 'had', 'a', 'lamb']

>>> [w.upper() for w in mary]
['MARY', 'HAD', 'A', 'LITTLE', 'LAMB']
>>> [len(w) for w in mary]
[4, 3, 1, 6, 4]
```

Same as `mary`

Filter out elements that do not meet a certain condition

Transform each element in list

Text processing on the fly

- Building a **python script** for a particular task is important.
- But before you are able to do that well, you should be comfortable **operating in IDLE shell**, exploring text files on the fly.
- We will try out, in **IDLE shell**:
  - Importing and using our text processing functions
  - Working with BIG text files

Import and use `textproc` in IDLE shell

- Using your **own module** in **IDLE shell**
  - Your module file can be imported in shell if it is in your **WD**.

```
>>> import textproc
Traceback (most recent call last):
. . .
ImportError: No module named textproc

>>> import os
>>> os.getcwd()
'D:\\Lab'
>>> os.chdir('text-processing')
>>> os.getcwd()
'D:\\Lab\\text-processing'

>>> import textproc
>>>
```

textproc.py cannot be found

Move into the directory where textproc.py is located

Now you can import the module

Import and use `textproc` in IDLE shell

```
>>> dir(textproc)
['__builtins__', '__doc__', '__file__',
'__name__', '__package__', 'getFreq',
'getRelFreq', 'getToks', 'getTypes', 'main',
'tale']

>>> help(textproc.getToks)
Help on function getToks in module textproc:

getToks(txt)
    Takes a text, returns a tokenized list of
words in lowercase

>>> textproc.getToks('Mary had a little lamb.')
['mary', 'had', 'a', 'little', 'lamb', '.']
```

See what's available in the module

Get help() on a function

Use getToks() function

Our text: Carroll's *Alice's Adventures*

- Download the novel from the Project Gutenberg web site: http://www.gutenberg.org/ebooks/11
  - Download and save the "Plain Text UTF-8" file.
  - Name it "alice.txt"

- Clean up the text file
  - Open it up using a text editor. You will find:
    - The file begins with a preamble
    - The file ends with many (over 300!) lines of Project Gutenberg Legalese

    ← Remove both parts and save.

Read in the text

```
>>> f = open(r'D:\Lab\text-processing\alice.txt')
>>> alicetxt = f.read()
>>> f.close()


>>> print alicetxt
```

STOP!!!
This sprays the *entire text* onto screen.

It will likely freeze your IDLE shell.

```
>>> print alicetxt[:500]

>>> print alicetxt[10000:10500]

>>> print alicetxt[-500:]
would feel with all their simple sorrows, and find a
pleasure in all their simple joys, remembering her own
child-life, and the happy summer days.
                THE END
```

Slice indexing is your friend.
Look at first 500 chars,
500 chars in the middle,
and last 500 characters.

Save your steps

- You WILL be crashing your IDLE through this exercise.
  - Ctrl + c might or might not help.

- Open up a text document and copy over your steps, starting from `textproc` importing
  - ← So you can quickly get back to where you were when you restart your session.

Tokenize the text

```
>>> alicetoks = textproc.getToks(alicetxt)
>>>
>>> alicetoks
```

DON'T press ENTER!!

This is a LONG list.

```
>>> alicetoks[100:150]
['well', 'as', 'she', 'could', ',', 'for',
'the', 'hot', 'day', 'made', 'her', 'feel',
'very', 'sleepy', 'and', 'stupid', ')',
',', 'whether', 'the', 'pleasure', 'of',
'making', 'a', 'daisy', '-', 'chain',
'would', 'be', 'worth', 'the', 'trouble',
'of', 'getting', 'up', 'and', 'picking',
'the', 'daisies', ',', 'when', 'suddenly',
'a', 'white', 'rabbit', 'with', 'pink',
'eyes', 'ran', 'close']
>>>
```

Again, use **slice indexing** to look at a portion at a time.

Explore the tokens

How long is the text?

How many times does 'rabbit' occur?

Does 'curiouser' occur in the text?

```
>>> len(alicetoks)
35399
>>> alicetoks.count('rabbit')
51
>>> 'curiouser' in alicetoks
True
```

How many of the word tokens are symbols?

```
>>> syms = [w for w in alicetoks if not w.isalnum()]
>>> len(syms)
8057
>>> set(syms)
set(['!', '"', "'", ')', '(', '*', '-', ',', '.', '--', '\xef\xbb\xbfalice', '_', ';', ':', ']', '[', '?'])
>>>
```

Unique word types

```
>>> alicetypes = textproc.getTypes(alicetoks)
>>>
>>> alicetypes
```

Nope!

```
>>> alicetypes[:30]
['!', '"', "'", '(', ')', '*', ',', '-', '--',
'.', '0', '3', ':', ';', '?', '[', ']', '_',
'a', 'abide', 'able', 'about', 'above',
'absence', 'absurd', 'acceptance', 'accident',
'accidentally', 'account', 'accounting']
>>>
```

Yep!

Explore the types

How many types?

What's the type token ratio?

```
>>> len(alicetypes)
2591
>>> len(alicetypes) /float(len(alicetoks))
0.07319415802706292
```

How many are 13+ characters long?

```
>>> [w for w in alicetypes if len(w) >=13]
['affectionately', 'circumstances', 'contemptuously',
'conversations', 'disappointment', 'extraordinary',
'inquisitively', 'multiplication', 'straightening',
'uncomfortable', 'uncomfortably']
>>> [(w, len(w)) for w in alicetypes if len(w) >=13]
[('affectionately', 14), ('circumstances', 13),
('contemptuously', 14), ('conversations', 13),
('disappointment', 14), ('extraordinary', 13),
('inquisitively', 13), ('multiplication', 14),
('straightening', 13), ('uncomfortable', 13),
('uncomfortably', 13)]
```

Careful with list comprehension

How many are 8 chars or longer?

```
>>> [w for w in alicetypes if len(w) >=8]
```

This is going to return a long list!

Unless you're reasonably sure your list is short, assign the list to a new variable first…

```
>>> foo = [w for w in alicetypes if len(w) >=8]

>>> len(foo)
627

>>> foo[:10]
['acceptance', 'accident', 'accidentally',
'accounting', 'accounts', 'accusation', 'accustomed',
'actually', 'addressed', 'addressing']
```

… and then look at snippets using **slice indexing**

Explore the types

How many types are vowel-less and 2+ characters?

```
>>> def hasVowel(w):
        return 'a' in w or 'e' in w or 'i' in w \
            or 'o' in w or 'u' in w


>>>
>>> [w for w in alicetypes if not hasVowel(w) and len(w) >=2]
['--', 'by', 'cry', 'dry', 'fly', 'hjckrrh', 'hm', 'll', 'my',
'sh', 'shy', 'shyly', 'sky', 'try', 'why']
>>>
```

Word frequencies

```
>>> alicefreq = textproc.getFreq(alicetoks)
>>> alicefreq['rabbit']
51
>>> alicefreq['the']
1644
>>> alicefreq['curiouser']
2

>>> alicefreq.keys()[:10]
['secondly', 'pardon', 'saves', 'knelt', 'four', 'sleep',
'hanging', 'ringlets', 'oldest', 'hate']

>>> alicefreq.items()[:10]
[('secondly', 2), ('pardon', 6), ('saves', 1), ('knelt', 1),
('four', 8), ('sleep', 6), ('hanging', 3), ('ringlets', 2),
('oldest', 1), ('hate', 2)]
>>>
```

Explore the frequencies

```
>>> for w in sorted(aflicereq, key=alicefreq.get, reverse=True)[:5] :
        print w, alicefreq[w]


' 2871
, 2418
the 1644
. 990
and 872
>>> once = [w for w in alicefreq if alicefreq[w] == 1]
>>> len(once)
1119
>>> once[:20]
['saves', 'knelt', 'oldest', 'blacking', 'inwards', 'sorry', 'rise',
'jack', 'seals', 'fireplace', 'prize', 'wooden', 'favoured',
'leaders', 'feathers', 'elegant', 'louder', 'machines', 'shining',
'hide']
```

What are the top 5 most frequent words?

How many types occur only once?

Wrap-up

- **Next class**
  - Working with a corpus
  - Pickling

- **Exercise #7**
  - List comprehension galore!
  - You will answer such exciting questions as:
    - What's the longest English word?
    - How many 5+ character palindromes are there?
    - What's the most common English letter to begin a word?