Homework 1: play with texts

1 Connect to our shared environment

In this assignment you will be playing with texts and learning a few basics of statistical NLP. Begin your work by connecting to GACRC's teaching cluster via `ssh` or PuTTY.

```
mylocalmachine $ ssh MyID@teach.gacrc.uga.edu
...
UGA DUO authentication is required for SSH/SCP access to
GACRC systems.

Duo two-factor login for MyID
Enter a passcode or select one of the following options:
 1. Duo Push to XXX-XXX-YYYY
...
Success. Logging you in..
[MyID@teach ~]$
```

If you are off-campus, you will need to use the VPN. Once you have authenticated using Duo, you are at the login node. Proceed to a compute node with `qlogin`

```
[MyID@teach ~]$ qlogin
[MyID@cX-YZ ~]
```

`qlogin` automatically chooses an interactive node for you; they have names like c2-4 or c2-5. The GACRC wiki has more details. But for now go ahead and set up for our class using the shell's `source` comand. Then when you start python all of the needed packages will be available.

```
[MyID@cX-YZ ~] source ling6570_config.sh
[yourUserID ~]$ python
Python 3.6.4 (default, May 24 2018, 13:05:12)
[GCC 6.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Please keep chapter 1 of the NLTK book handy, perhaps open in another browser window. You do **not** need to re-download the NLTK corpora. Those and more are already set up on our shared environment. To get started, you **do** need to issue:

```
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
```

```
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

This import command sets up your exploration; you can always type `texts()` to be reminded of what is available.

1.1 the python language

For more on python itself, take a look at the official tutorial or Na-Rae Han's excellent python3 notes which are mirrored on our eLC. These includes step-by-step videos!

2 Hello, texts

Consider `text4`, the inaugural addresses by US Presidents. Where does the word "government" appear?

```
text4.concordance('government')
```

```
Displaying 25 of 600 matches:
the people of the United States a Government instituted by themselves for thes
hed in the system of their united government the tranquil deliberations and vo
the proceedings of a new and free government can more auspiciously commence .
ity , and the preeminence of free government be exemplified by all the attribu
estiny of the republican model of government are justly considered , perhaps ,
nefits of an united and effective government , or which ought to await the fut
paralleled unanimity on a form of government for the security of their union a
ures on which the success of this Government must depend . Fellow citizens , I
d during my administration of the Government I have in any instance violated w
nly arise concerning the forms of government to be instituted over the whole a
nary war , supplying the place of government , commanded a degree of order suf
a courier may go from the seat of government to the frontier in a single day ,
the present happy Constitution of Government . Employed in the service of my c
s conformable to such a system of government as I had ever most esteemed , and
ation for it . What other form of government , indeed , can so well deserve ou
 other Chamber of Congress , of a Government in which the Executive authority
 their good , in every legitimate government , under whatever form it may appe
 appear . The existence of such a government as ours for any length of time is
ough artifice or corruption , the Government may be the choice of a party for
or , intrigue , or venality , the Government may not be the choice of the Amer
amiable and interesting system of government ( and such are some of the abuses
 principle , of a free republican government , formed upon long and serious re
pe which has been adopted by this Government and so solemnly sanctioned by bot
res the honor and interest of the Government and its constituents demand ; if
blessing upon this nation and its Government and give it all possible success
```

Q1 Look up the words "fathers". What words typically go alongside it? Now compare to "father" i.e. singular.

How are its contexts different? Try a few others as suggested in §1.3 of NLTK book chapter 1 and turn in a paragraph reflecting on what you find.

Section 1.3 of chapter 1 explores the word "monstrous". Which book uses "monstrous" more, Moby Dick or Sense and Sensibility ?

```
text2.count('monstrous')
```

```
11
```

```
text1.count('monstrous')
```

```
10
```

Each book can be viewed as a frequency distribution, tallying up how often a particular word appears. You can represent this in the computer by instantiating an object of class `FreqDist`, initializing it with a given text. (For more on object-oriented programming, see Bernd Klein's tutorial)

```
fdist4 = FreqDist(text4)
```

If you have X11 graphics set up, you can view this frequency distribution

```
fdist4.plot(50,title="Inaugural")
```

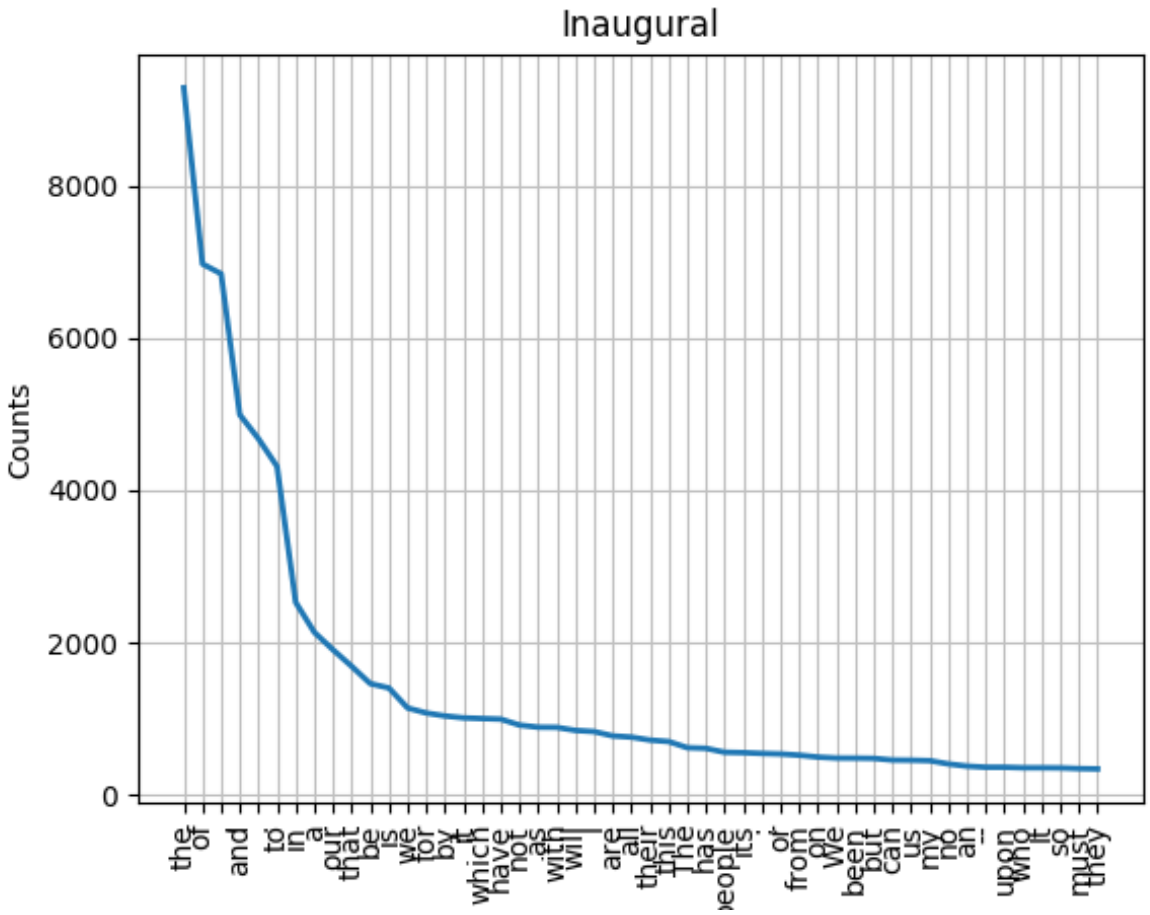

Figure 1: fdist4

Without X11, you can still view it on your terminal using the `tabulate` function. Take a look at tabulate's docstring:

```
help(fdist4.tabulate)
```

and see that the argument to the `tabulate` function is the number of samples, i.e. the number of top-ranked words whose frequencies you would like to know.

```
fdist4.tabulate(15)
```

```
 the   of    ,  and    .   to   in    a  our that   be   is   we  for   by
9446 7087 7045 5146 4856 4414 2561 2184 2021 1748 1483 1448 1211 1110 1053
```

How many **types** versus **tokens** are there in these Inaugural Addresses?

```
len(set(text4))
```

```
9913
```

```
len(text4)
```

```
149797
```

The expression `set(text4)` forms a set that is exactly the vocabulary of `text4`. Both sets and nltk.texts are "containers" to which the len function can apply.

3 Frequency Distributions

Now lets form a frequency distribution for Sense and Sensibility and use list comprehensions to ask how many words have a particular property:

```
austen = FreqDist(text2)
# use a list comprehension to compare the number of hapaxes
len([word for word in austen if austen[word] == 1])
```

```
2694
```

```
# or simply  len(austen.hapaxes())
#
#
# to the number of words attested 2 times
len([word for word in austen if austen[word] == 2])
```

```
994
```

Q2 In the example, we've considered attestation counts x at $x = 1$ and $x = 2$. What about other levels of x? How does the number of words attested at a particular count change as that count gets larger? Try a few values and see if you can come up with a generalization.

Q3 Read §3.2 of the NLTK book's first chapter and then craft a list comprehension that picks out words in Jane Austen that are attested more than ten times and end in *-ly*

Now lets form another frequency distribution, one in which we first clean-up the words of Sense and Sensibility to ensure that they're not entirely punctuation characters, and downcased so that we count case-variants as the same word.

```
from string import punctuation
f = FreqDist(w.lower() for w in text2.tokens if all(c not in punctuation for c in w))
```

Read about the `all()` function in your python 3.6 docs under "Library Reference" part 2 "Built-in Functions" and Na-Rae Han's python3 Additional Topic 15. Use the `most_common` method of this frequency distribution to get a list of pairs; this is how `tabulate` works. Pairs are 2-tuples; if you are unfamilar with tuples in python, read up on them in the official tutorial.

```
f.most_common(50)
```

Q4 What proper names are well-attested in Sense and Sensibility ? Use the `similar` function to ask which other words occur in the same distributional contexts as those proper names. Turn in a characterization of these words as a class.

To learn what `similar` does, use the `help` function to display its docstring.

```
help(text2.similar)
```

These distributional contexts are just single words on either side of the target word. You can view them by making a concordance.

Q5 Same question as Q4 but now about "husband" instead. Can you discern a different natural class in the output of `similar('husband')` ?

4 Bigrams

As introduced in §3.3 of chapter 1 of the NLTK book, pairs of adjacent words are called bigrams. This is a case of the more general notion of n gram where $n = 2$ The bigrams of a sentence can be printed out in a for-loop, introduced in §4.3 and given the video treatment in NRH's Tutorial 14.

```
for b in bigrams(sent1):
   print(b)
```

```
('Call', 'me')
('me', 'Ishmael')
('Ishmael', '.')
```

Bigrams can serve as the support of a frequency distribution:

```
bd = FreqDist(bigrams(text4))
bd.most_common(10)
```

```
[(('of', 'the'), 1759),
 ((',', 'and'), 1331),
 (('in', 'the'), 748),
 (('to', 'the'), 702),
 (('of', 'our'), 618),
 (('.', 'The'), 591),
 (('.', 'We'), 514),
 (('and', 'the'), 460),
 ((',', 'the'), 373),
 (('.', 'It'), 346)]
```

Q6 What's the most popular word preceding 'whale' in Moby Dick? Use bigrams to initialize a `FreqDist`. In doing so, consider using a list comprehension or perhaps a generator expression (described further in NLTK §4.2).

Then you can use one of FreqDist's handy methods to read off the answer. These are summarized in Table 3.1 of NLTK chapter 1.

Imagine winnowing these bigrams down to just those that are "interesting"

```
text1.collocations()
```

```
Sperm Whale; Moby Dick; White Whale; old man; Captain Ahab; sperm
whale; Right Whale; Captain Peleg; New Bedford; Cape Horn; cried Ahab;
years ago; lower jaw; never mind; Father Mapple; cried Stubb; chief
mate; white whale; ivory leg; one hand
```

One way to quantify "interestingness" is via log-likelihood ratios. This described in section 5.3.4 of Manning and Schuetze 1999 and implemented in NLTK's BigramAssocMeasures object. Note that English stopwords are excluded for you by this high-level function.

Finally, generate some new random text. Here, first created a new `nltk.text` based on a cleaned-up version of `text1`.

```
alttext1 = Text(w.lower() for w in text1.tokens if all(c not in punctuation for c in w))
alttext1.generate()
```

As you can see for yourself via `help`, this `generate` function is based on trigrams — three-word sequences that govern the probabilities with which successive words are emitted.

Q7 Critique the trigram model. What aspect of English are missing or poorly-modeled, based on your experience with this `generate` function?