Jupyter Notebook for Week 3

Student Name: Muhammad Rakibul Islam

- Date:
- Assignment Due:
- Instructor: Lisa Rhody
- Methods of Text Analysis, Spring 2023

For the following assignment, you will be working through exercises from *Natural Language Processing with Python: Analyzing Text with the Natural Langauge Toolkit* by Steven Bird, Ewan Klein, and Edward Loper. The book can be found at http://www.nltk.org/book/. For those students who are working on their own workbooks using Jupyter and GitHub, it might be helpful to have this window open on 1/2 of your computer monitor and the text of the NLP with Python book open right next to it on the other 1/2 of your screen.

For those of you who are using Google Colab, this asignment includes a few extra steps in order to have things work in the Colab environment, so you will not be able to follow the book exactly.

Periodically, throughout the notebook, you will be prompted to add a "markdown" cell and to write a reflection that connects your secondary reading with the code that you are running in the Jupyter notebook.

I encourage you to add cells and to create little experiments as you go. If you do, please alert me to them by inserting a markdown cell directly above or below your experiment to explain what you are doing.

Getting Started

If you are working on your own computer, you'll want to download NLTK on your local machine. There are instructions on how to do this on the NLTK site (https://www.nltk.org/install.html). You can also use the Anaconda Distribution (https://www.anaconda.com/distribution). Once you install the NLTK package, you'll need to install necessary datasets/models for future assignments to work.

Working with NLTK

In order for us to work with NLTK, we need to load it into Google Colab (or Jupyter or the Python interpreter). We do this by first importing NLTK and then identifying what parts of NLTK we want to work with. Once the data is added to your working environment, you can load some into the Google Colab environment.

```
## first import the NLTK package
import nltk
## then use Python to download the "book" section of NLTK
nltk.download("book")
```

```
[nltk_data] Downloading collection 'book'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dependency_treebank.zip.
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/genesis.zip.
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/gutenberg.zip.
[nltk_data]    | Downloading package ieer to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ieer.zip.
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/inaugural.zip.
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/movie_reviews.zip.
[nltk_data]    | Downloading package nps_chat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/nps_chat.zip.
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/names.zip.
[nltk_data]    | Downloading package ppattach to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ppattach.zip.
[nltk_data]    | Downloading package reuters to /root/nltk_data...
[nltk_data]    | Downloading package senseval to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/senseval.zip.
[nltk_data]    | Downloading package state_union to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/state_union.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/stopwords.zip.
[nltk_data]    | Downloading package swadesh to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/swadesh.zip.
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/timit.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
```

```
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/toolbox.zip.
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr.zip.
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/webtext.zip.
```

When you have run the above cell correctly, the last two lines should read "Done Downloading collection book and then "True."

In this next part, you're going to use Python to tell the computer "from NLTK's book module, load all items." The book module contains the data that you'll need for this chapter, which is a corpus of book text that has been pre-prepared for use with NLTK. It's important to know that these texts are already cleaned for you, as it will be much neater and easier to work with than if you were working with another plain text file that was not prepared in advance. We'll look at files that have not been prepared later, but for now, let's work with the files that come as part of NLTK.

```
## We are telling Colab to download all the data inside the book package in NLTK.
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
```

We can test the success of our import by simply calling up the label that has been applied to the texts. In this case, the texts are labeled text1, text2, text3, etc.

```
text1
```

```
<Text: Moby Dick by Herman Melville 1851>
```

```
text2
```

```
<Text: Sense and Sensibility by Jane Austen 1811>
```

By importing the NLTK book package and then the book corpora, we have access to 9 different full text books. In the next sections, we're going to search, count, and manipulate texts using NLTK.

Searching Text

```
text1.concordance("monstrous")
```

```
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
```

REFLECTION: In *The Textual Condition*, McGann writes: "Both the practice and the study of human culture comprise a network of symbolic exchanges. Because human beings are not angels, these exchanges always involve material negotiations." What is the "symbolic exchange" that happens in creating a concordance like the one above? How are these "material negotiations" or not? Consider how this may or may not relate to the use of the method "similar" below.

Response:

The minute we are transforming part of the text into a format that allows us to perform analysis (in this case listing how how many times and the context in which the word "monstrous" appears), we are making the decision to choose which word(s) to include, how to present them etc. Not only that we want to present some form of meaning and interpretation which to me is the "symbolic exhange" that occurs here.

These are material negotiations because at the end of the day, somebody had to literally negotiate with the publication company that allows us to utilize the text in the first place. They required funding and had to spend time and resources to put this together.

On my end, I had to also put in labor and capital and other resources (my laptop, the internet etc.) to come to the stage where I am creating this concordance.

As such, I do believe material negotations are involved.

Also, symbolic exhange applies to the use of the method "similar" below because again we are making the decision to interpret what other text is similar to "monstrous" thus making our own meaning to the matter.

```
text1.similar("monstrous")
```

```
true contemptible christian abundant few part mean careful puzzled
mystifying passing curious loving wise doleful gamesome singular
delightfully perilous fearless
```

```
text2.common_contexts(["monstrous", "very"])
```

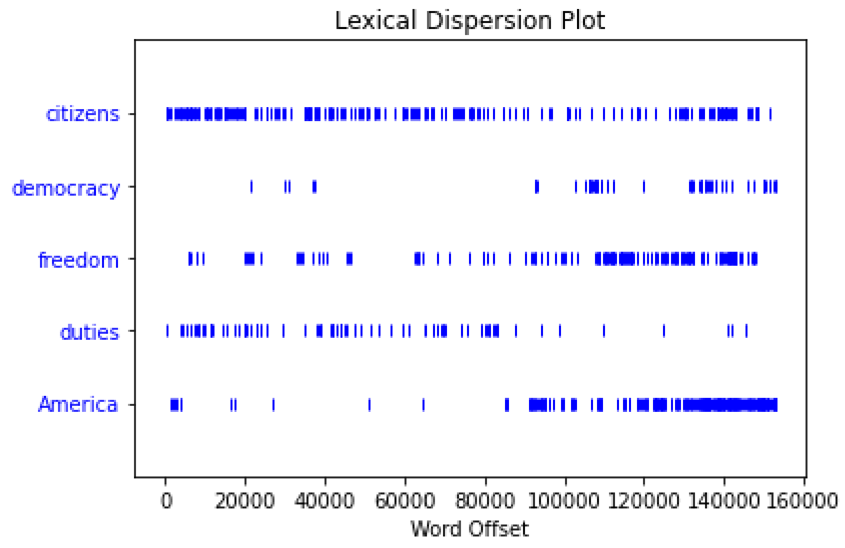
```
am_glad a_pretty a_lucky is_pretty be_glad
```

REFLECTION: How does your sense of "context" and "symbolic exchange" change (or not) when you view the frequency of a word's use distributed across a novel using a "dispersion plot."?

Response:

I guess the way that my sense of "context" and symbolic exchange" changes viewing a dispersion plot is that the plot shows the density of the occurence of certain words throughtout the course of the text which can give rise to certain themes being prominent in a section (and not when when less existent) and as a result, I as the analyst can make interpretations based on these frequences and give my own meaning which builds my sense of "context" and "symbolic exchange" in this case.

```
text4.dispersion_plot(["citizens", "democracy", "freedom", "duties", "America"])
```



1.4 Counting Vocabulary

Follow along in http://www.nltk.org/book/ch01.html by reading the description and then running the cells below. What do the authors mean by a "token"? How is a "token" different from "text"?

Response:

A "token" is being defined as *a* sequence of characters that represents a unit of meaning in a text, while "text" refers to the entire body of words that are being analyzed.

```
len(text3)
```

```
    44764
```

```
## test
len(text4)
```

```
    152901
```

```
## note to self : this helps find the distinct word / punctuation symbols count
sorted(set(text3))
```

```
    ['!',
     "'",
     '(',
     ')',
     ',',
     ',)',
     '.',
     '.)',
     ':',
     ';',
     ';)',
     '?',
     '?)',
     'A',
     'Abel',
     'Abelmizraim',
     'Abidah',
     'Abide',
     'Abimael',
     'Abimelech',
     'Abr',
     'Abrah',
     'Abraham',
     'Abram',
     'Accad',
     'Achbor',
     'Adah',
     'Adam',
     'Adbeel',
     'Admah',
     'Adullamite',
     'After',
     'Aholibamah',
     'Ahuzzath',
     'Ajah',
     'Akan',
     'All',
     'Allonbachuth',
     'Almighty',
     'Almodad',
     'Also',
     'Alvah',
     'Alvan',
     'Am',
     'Amal',
     'Amalek',
     'Amalekites',
     'Ammon',
     'Amorite',
     'Amorites',
     'Amraphel',
     'An',
     'Anah',
     'Anamim',
     'And',
     'Aner',
     'Angel',
     'Appoint',
```

```
len(set(text3))
```

```
    2789
```

What is the difference between "word types" and a "types"? How are these meanings distinct from other uses of the word "types" that you are familiar with?

Response:

"Word types" refers to the total number of unique words in a text or corpus.

"Types" can refer to not only word types, but also other linguistic elements like punctuations.

I actually think they are similar to the use of the word types that I would otherwise use.

```
## measure of the lexical richness of the text (percentage of unique to total words /
len(set(text3)) / len(text3)
```

```
    0.06230453042623537
```

```
## how many times a particular word shows up in the text
text3.count("smote")
```

```
    5
```

```
## finds percentage of the text taken up by a certain word
100 * text4.count('a') / len(text4)
```

```
    1.457806031353621
```

```
## How many times does the word lol appear in text5
text5.count("lol")
```

```
    704
```

```
##How much is this as a percentage of the total number of words in this text?
100 * text5.count('lol') / len(text5)
```

```
    1.5640968673628082
```

```
##instead of retyping a particular set of text a function can be used. Test below.

def lexical_diversity(text):
    return len(set(text)) / len(text)
def percentage(count, total):
    return 100 * count / total
```

```
lexical_diversity(text3)
```

```
    0.06230453042623537
```

```
lexical_diversity(text5)
```

```
    0.13477005109975562
```

```
percentage(4, 5)
```

```
    80.0
```

```
percentage(text4.count('a'), len(text4))
```

```
    1.457806031353621
```

REFLECTION: McGann writes on page 4 that "Today, texts are largely imagined as scenes of reading rather than scenes of writing. This 'readerly' view of text has been most completely elaborated through the modern hermeneutical tradition in which text is not something we *make* but something we *interpret*." How does McGann's view of the text relate to the "reading" that happens in this notebook? Consider his later question: "Where--and how--does the activity of reading take place?"

Response:

In this notebook, McGann's view of the text relates through the interaction between me, the user, and the responses that are generated by python. While the responses are being generated by a machine, the choice of what analysis to perform and finally the interpretation that is made is being done by me, a human reader thus giving it a sense of meaning through the mind of the reader aligning with the "'readerly' view of text".

As for the last question, again given the context of this notebook, the activity of this reading takes place in two two place. First is within the technological framework in which this is being done as well as the haman cognitive processes such as reasoning and memory that I am utilizing to write code and finally analyze and interpret the results.

Chapter 2 Accessing Text Corpora and Lexical Resources

Ok, now we're going to shift a bit and look at Chapter 2 of the NLP book http://www.nltk.org/book/ch02.html. (Don't worry, we'll come back to the rest of Chapter 1 next week.) McGann talks about different *types* of text in his introduction: "Even the most 'informational' text comprises an interactive mechanism of communicative exchange." We're going to look at different texts in the NLTK corpus. Again, the easiest way to proceed here is to open Chapter 2 in a window on one side of your monitor and this notebook in the other and to read simultaneously.

```
# remember, we already ran import nltk earlier in the notebook
nltk.corpus.gutenberg.fileids()
```

```
    ['austen-emma.txt',
     'austen-persuasion.txt',
     'austen-sense.txt',
     'bible-kjv.txt',
     'blake-poems.txt',
     'bryant-stories.txt',
     'burgess-busterbrown.txt',
     'carroll-alice.txt',
     'chesterton-ball.txt',
     'chesterton-brown.txt',
     'chesterton-thursday.txt',
     'edgeworth-parents.txt',
     'melville-moby_dick.txt',
     'milton-paradise.txt',
     'shakespeare-caesar.txt',
     'shakespeare-hamlet.txt',
     'shakespeare-macbeth.txt',
     'whitman-leaves.txt']
```

```
#choosing Emma by Jane Austen and calling it emma in short for ease
emma = nltk.corpus.gutenberg.words('austen-emma.txt')
len(emma)
```

```
192427
```

```
emma = nltk.Text(nltk.corpus.gutenberg.words('austen-emma.txt'))
emma.concordance("surprize")
```

```
Displaying 25 of 37 matches:
er father , was sometimes taken by surprize at his being still able to pity `
hem do the other any good ." " You surprize me ! Emma must do Harriet good : a
Knightley actually looked red with surprize and displeasure , as he stood up ,
r . Elton , and found to his great surprize , that Mr . Elton was actually on
d aid ." Emma saw Mrs . Weston ' s surprize , and felt that it must be great ,
father was quite taken up with the surprize of so sudden a journey , and his f
y , in all the favouring warmth of surprize and conjecture . She was , moreove
he appeared , to have her share of surprize , introduction , and pleasure . Th
ir plans ; and it was an agreeable surprize to her , therefore , to perceive t
talking aunt had taken me quite by surprize , it must have been the death of m
f all the dialogue which ensued of surprize , and inquiry , and congratulation
 the present . They might chuse to surprize her ." Mrs . Cole had many to agre
the mode of it , the mystery , the surprize , is more like a young woman ' s s
 to her song took her agreeably by surprize -- a second , slightly but correct
" " Oh ! no -- there is nothing to surprize one at all .-- A pretty fortune ;
t to be considered . Emma ' s only surprize was that Jane Fairfax should accep
of your admiration may take you by surprize some day or other ." Mr . Knightle
ation for her will ever take me by surprize .-- I never had a thought of her i
 expected by the best judges , for surprize -- but there was great joy . Mr .
 sound of at first , without great surprize . " So unreasonably early !" she w
d Frank Churchill , with a look of surprize and displeasure .-- " That is easy
; and Emma could imagine with what surprize and mortification she must be retu
tled that Jane should go . Quite a surprize to me ! I had not the least idea !
 . It is impossible to express our surprize . He came to speak to his father o
g engaged !" Emma even jumped with surprize ;-- and , horror - struck , exclai
```

```
from nltk.corpus import gutenberg
gutenberg.fileids()
```

```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

```
emma = gutenberg.words('austen-emma.txt')
```

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
#program displays three statistics for each text: average word length, average sentenc
for fileid in gutenberg.fileids():
    num_chars = len(gutenberg.raw(fileid))
    num_words = len(gutenberg.words(fileid))
    num_sents = len(gutenberg.sents(fileid))
    num_vocab = len(set(w.lower() for w in gutenberg.words(fileid)))
    print(round(num_chars/num_words), round(num_words/num_sents), round(num_words/num_
```

```
5 25 26 austen-emma.txt
5 26 17 austen-persuasion.txt
5 28 22 austen-sense.txt
4 34 79 bible-kjv.txt
5 19 5 blake-poems.txt
4 19 14 bryant-stories.txt
4 18 12 burgess-busterbrown.txt
4 20 13 carroll-alice.txt
5 20 12 chesterton-ball.txt
5 23 11 chesterton-brown.txt
5 19 11 chesterton-thursday.txt
4 21 25 edgeworth-parents.txt
5 26 15 melville-moby_dick.txt
5 52 11 milton-paradise.txt
4 12 9 shakespeare-caesar.txt
4 12 8 shakespeare-hamlet.txt
```

```
4 12 7 shakespeare-macbeth.txt
5 36 12 whitman-leaves.txt
```

```
macbeth_sentences = gutenberg.sents('shakespeare-macbeth.txt')
macbeth_sentences
```

```
[['[', 'The', 'Tragedie', 'of', 'Macbeth', 'by', 'William', 'Shakespeare',
'1603', ']'], ['Actus', 'Primus', '.'], ...]
```

1.2 Web and Chat Text

```
from nltk.corpus import webtext
for fileid in webtext.fileids():
    print(fileid, webtext.raw(fileid)[:65], '...')
```

```
firefox.txt Cookie Manager: "Don't allow sites that set removed cookies to se ..
grail.txt SCENE 1: [wind] [clop clop clop]
KING ARTHUR: Whoa there!  [clop ...
overheard.txt White guy: So, do you have any plans for this evening?
Asian girl ...
pirates.txt PIRATES OF THE CARRIBEAN: DEAD MAN'S CHEST, by Ted Elliott & Terr ..
singles.txt 25 SEXY MALE, seeks attrac older single lady, for discreet encoun ..
wine.txt Lovely delicate, fragrant Rhone wine. Polished leather and strawb ...
```

```
from nltk.corpus import nps_chat
chatroom = nps_chat.posts('10-19-20s_706posts.xml')
chatroom[123]
```

```
['i',
 'do',
 "n't",
 'want',
 'hot',
 'pics',
 'of',
 'a',
 'female',
 ',',
 'I',
 'can',
 'look',
 'in',
 'a',
 'mirror',
 '.']
```

```
from nltk.corpus import brown
brown.categories()
```

```
['adventure',
 'belles_lettres',
 'editorial',
 'fiction',
 'government',
 'hobbies',
 'humor',
 'learned',
 'lore',
 'mystery',
 'news',
 'religion',
 'reviews',
 'romance',
 'science_fiction']
```

1.3 Brown Corpus

```
#count of modal verbs for a particular genrre. Helpful for studying systematic differe
from nltk.corpus import brown
news_text = brown.words(categories='news')
fdist = nltk.FreqDist(w.lower() for w in news_text)
modals = ['can','could','may','might','must','will']
for m in modals:
    print(m + ":", fdist[m], end=" ")
```

```
can: 94 could: 87 may: 93 might: 38 must: 53 will: 389
```

```
#Counting a selection of wh words from another section
from nltk.corpus import brown
fiction_text = brown.words(categories='fiction')
fdist = nltk.FreqDist(w.lower() for w in fiction_text)
modals = ['what','when','where','who','why']
for m in modals:
    print(m + ":", fdist[m], end=" ")
```

```
what: 186 when: 192 where: 89 who: 112 why: 42
```

```
brown.sents(categories=['news','editorial','reviews'])
```

```
[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an',
'investigation', 'of', "Atlanta's", 'recent', 'primary', 'election', 'produced',
'``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'place',
'.'], ['The', 'jury', 'further', 'said', 'in', 'term-end', 'presentments',
'that', 'the', 'City', 'Executive', 'Committee', ',', 'which', 'had', 'over-
all', 'charge', 'of', 'the', 'election', ',', '``', 'deserves', 'the', 'praise',
'and', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta', "''", 'for', 'the',
'manner', 'in', 'which', 'the', 'election', 'was', 'conducted', '.'], ...]
```

```
#Counts for each genre of interest using NLTK's support for conditional frequency dist
cfd = nltk.ConditionalFreqDist(
        (genre,word)
        for genre in brown.categories()
        for word in brown.words(categories=genre))
genres = ['news', 'relgion', 'hobbies', 'science_fiction', 'romance', 'humor']
modals = ['can', 'could', 'may', 'might', 'must', 'will']
cfd.tabulate(conditions=genres, samples=modals)
```

```
                  can could   may might  must  will
           news    93    86    66    38    50   389
        relgion     0     0     0     0     0     0
        hobbies   268    58   131    22    83   264
science_fiction    16    49     4    12     8    16
        romance    74   193    11    51    45    43
          humor    16    30     8     8     9    13
```

```
from nltk.corpus import inaugural
inaugural.fileids()
```

```
['1789-Washington.txt',
 '1793-Washington.txt',
 '1797-Adams.txt',
 '1801-Jefferson.txt',
 '1805-Jefferson.txt',
 '1809-Madison.txt',
 '1813-Madison.txt',
 '1817-Monroe.txt',
 '1821-Monroe.txt',
 '1825-Adams.txt',
 '1829-Jackson.txt',
 '1833-Jackson.txt',
 '1837-VanBuren.txt',
 '1841-Harrison.txt',
 '1845-Polk.txt',
 '1849-Taylor.txt',
 '1853-Pierce.txt',
 '1857-Buchanan.txt',
 '1861-Lincoln.txt',
 '1865-Lincoln.txt',
 '1869-Grant.txt',
 '1873-Grant.txt',
 '1877-Hayes.txt',
 '1881-Garfield.txt',
 '1885-Cleveland.txt',
 '1889-Harrison.txt',
 '1893-Cleveland.txt',
 '1897-McKinley.txt',
 '1901-McKinley.txt',
 '1905-Roosevelt.txt',
 '1909-Taft.txt',
 '1913-Wilson.txt',
 '1917-Wilson.txt',
 '1921-Harding.txt',
 '1925-Coolidge.txt',
 '1929-Hoover.txt',
 '1933-Roosevelt.txt',
 '1937-Roosevelt.txt',
 '1941-Roosevelt.txt',
 '1945-Roosevelt.txt',
 '1949-Truman.txt',
 '1953-Eisenhower.txt',
 '1957-Eisenhower.txt',
 '1961-Kennedy.txt',
 '1965-Johnson.txt',
 '1969-Nixon.txt',
 '1973-Nixon.txt',
 '1977-Carter.txt',
 '1981-Reagan.txt',
 '1985-Reagan.txt',
 '1989-Bush.txt',
 '1993-Clinton.txt',
 '1997-Clinton.txt',
 '2001-Bush.txt',
 '2005-Bush.txt',
 '2009-Obama.txt',
 '2013-Obama.txt',
 '2017-Trump.txt',
```

```
from nltk.corpus import reuters
reuters.fileids()
```

```
     ['test/14826',
      'test/14828',
      'test/14829',
      'test/14832',
      'test/14833',
      'test/14839',
      'test/14840',
      'test/14841',
      'test/14842',
      'test/14843',
      'test/14844',
      'test/14849',
      'test/14852',
      'test/14854',
      'test/14858',
      'test/14859',
      'test/14860',
      'test/14861',
      'test/14862',
      'test/14863',
      'test/14865',
      'test/14867',
      'test/14872',
      'test/14873',
      'test/14875',
      'test/14876',
      'test/14877',
      'test/14881',
      'test/14882',
      'test/14885',
      'test/14886',
      'test/14888',
      'test/14890',
      'test/14891',
      'test/14892',
      'test/14899',
      'test/14900',
      'test/14903',
      'test/14904',
      'test/14907',
      'test/14909',
      'test/14911',
      'test/14912',
      'test/14913',
      'test/14918',
      'test/14919',
      'test/14921',
      'test/14922',
      'test/14923',
      'test/14926',
      'test/14928',
      'test/14930',
      'test/14931',
      'test/14932',
      'test/14933',
      'test/14934',
      'test/14941',
```

```
reuters.categories()
```

```
['acq',
 'alum',
 'barley',
 'bop',
 'carcass',
 'castor-oil',
 'cocoa',
 'coconut',
 'coconut-oil',
 'coffee',
 'copper',
 'copra-cake',
 'corn',
 'cotton',
 'cotton-oil',
 'cpi',
 'cpu',
 'crude',
 'dfl',
 'dlr',
 'dmk',
 'earn',
 'fuel',
 'gas',
 'gnp',
 'gold',
 'grain',
 'groundnut',
 'groundnut-oil',
 'heat',
 'hog',
 'housing',
 'income',
 'instal-debt',
 'interest',
 'ipi',
 'iron-steel',
 'jet',
 'jobs',
 'l-cattle',
 'lead',
 'lei',
 'lin-oil',
 'livestock',
 'lumber',
 'meal-feed',
 'money-fx',
 'money-supply',
 'naphtha',
 'nat-gas',
 'nickel',
 'nkr',
 'nzdlr',
 'oat',
 'oilseed',
 'orange',
 'palladium',
```

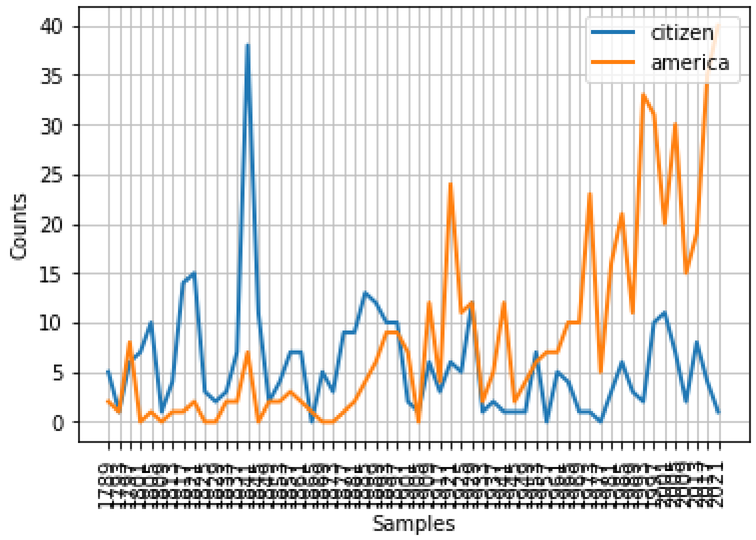
You can skip the remainder of the Reuters section. We'll return to this later.

1.5 Inaugural Address Corpus

```
from nltk.corpus import inaugural
inaugural.fileids()
```

```
['1789-Washington.txt',
 '1793-Washington.txt',
 '1797-Adams.txt',
 '1801-Jefferson.txt',
 '1805-Jefferson.txt',
 '1809-Madison.txt',
 '1813-Madison.txt',
 '1817-Monroe.txt',
 '1821-Monroe.txt',
 '1825-Adams.txt',
 '1829-Jackson.txt',
 '1833-Jackson.txt',
 '1837-VanBuren.txt',
 '1841-Harrison.txt',
 '1845-Polk.txt',
 '1849-Taylor.txt',
 '1853-Pierce.txt',
 '1857-Buchanan.txt',
 '1861-Lincoln.txt',
 '1865-Lincoln.txt',
 '1869-Grant.txt',
 '1873-Grant.txt',
 '1877-Hayes.txt',
 '1881-Garfield.txt',
 '1885-Cleveland.txt',
 '1889-Harrison.txt',
 '1893-Cleveland.txt',
 '1897-McKinley.txt',
 '1901-McKinley.txt',
 '1905-Roosevelt.txt',
 '1909-Taft.txt',
 '1913-Wilson.txt',
 '1917-Wilson.txt',
 '1921-Harding.txt',
 '1925-Coolidge.txt',
 '1929-Hoover.txt',
 '1933-Roosevelt.txt',
 '1937-Roosevelt.txt',
 '1941-Roosevelt.txt',
 '1945-Roosevelt.txt',
 '1949-Truman.txt',
 '1953-Eisenhower.txt',
 '1957-Eisenhower.txt',
 '1961-Kennedy.txt',
 '1965-Johnson.txt',
 '1969-Nixon.txt',
 '1973-Nixon.txt',
 '1977-Carter.txt',
 '1981-Reagan.txt',
 '1985-Reagan.txt',
 '1989-Bush.txt',
 '1993-Clinton.txt',
 '1997-Clinton.txt',
 '2001-Bush.txt',
 '2005-Bush.txt',
 '2009-Obama.txt',
 '2013-Obama.txt',
 '2017 Trump txt'
```

```
#Plot of a Conditional Frequency Distribution. Count of words in the Inaugural Address
#Words in the corpus converted to lowercase using w.lower() in the code
#startswith() checks if they start with 'america' or 'citizen'
cfd = nltk.ConditionalFreqDist(
        (target, fileid[:4])
        for fileid in inaugural.fileids()
        for w in inaugural.words(fileid)
        for target in ['america', 'citizen']
        if w.lower().startswith(target))
cfd.plot()
```



```
<AxesSubplot:xlabel='Samples', ylabel='Counts'>
```

1.6 Annotated Text Corpora

[ ] ↳ *35 cells hidden*

Colab paid products - Cancel contracts here