Applications of Artificial Intelligence

EAI 6010, CRN 70749

Professor Vladimir Shapiro

Module 3: Assignment Week 3 - NLP AI Applications

Submitted By - Richa Umesh Rambhia

NLP AI Applications - Project Gutenberg & Inaugural Corpus

Table of Contents

Introduction

Natural Language Processing is the process that applies various concepts and techniques in order to process, analyze, and interpret the human language. The various techniques and methodologies used in NLP are *tokenization, stemming, lemmatization, stop-word removal, n-grams, named entity recognition, and part-of-speech tagging*.

NLTK, i.e, the Natural Language Toolkit is one such platform of NLP that provides to build Python programs that could work with human language data. [1] The toolkit has about **50 corpora and lexical resources** that uses wordnet along with the various concepts, techniques, and processing libraries for classification, tokenization, stemming, tagging, parsing, etc. [1]

A corpus on the other hand is defined as a collection of various text documents which can together be thought as a bunch of text. [2] The corpus that is used for this task is the **Gutenberg corpus** which has the texts of 25000 electronic books extracted from the website [3], and the **Inaugural corpus** consists text of the US presidents inaugural addresses since 1789. [4]

The aim of this assignment is to analyze the two corpus using NLTK in order to explore the data and answer some of the questions with respect to *word count, frequency, finding the Synonyms and Hyponyms of the words* using **WordNet**. This will give an overall idea of how Natural Language Processing can be used to understand human language and analyze various textual data using the NLP techniques.

Analysis

This assignment deals with the analysis of the two corpus data in NLTK, i.e, the *Gutenberg corpus and the Inaugural corpus.* As mentioned above, a text corpus is a large set of texts for which specific data is extracted from the corpus in order to analyze the data.

Considering the Guternberg corpus, NLTK consists of this corpus which has a small amount of texts extracted from the Project Gutenberg electronic text archive. [3] Accessing each of the text in the corpus at an individual level for analysis, is what the task majorly deals with, along with the analysis of the relative frequencies of the specified modals present in the text corpus. With the help of the various functions available for the gutenberg corpus in NLTK's corpus package that is downloaded, the analysis becomes much easier.

Similar to the gutenberg corpus, considering the Inaugural corpus, which consists of the text of the US presidents addressess, the words in each of the text are extracted along with the count frequency of the words in order to extract the most commonly used top ten words. The synonyms and hyponyms of these top 10 words or most commonly used words are printed in order to extract the word which has the largest number of synonyms and hyponyms.

Gutenberg Corpus

Q1:

a. Download and install the Gutenberg corpus tool to your Jupyter Notebook. Project Gutenberg contains some 25,000 free electronic books hosted at http://www.gutenberg.org/ (http://www.gutenberg.org/). We can install the NLTK package, then use the Gutenberg corpus in it. It can be installed by running the following in the computer terminal:

A1.

a. Installing packages

In [ ]:
```
!pip install nltk
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheel
s/public/simple/ (https://us-python.pkg.dev/colab-wheels/public/simple/)
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.7)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from nltk) (7.1.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.7/dist-packages (from nltk) (2022.6.
2)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from nltk) (1.2.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from nltk) (4.64.1)
```

Q1:

b. Download the Gutenberg corpus tool in the NLTK package by, e.g.:

A1. b. Downloading corpus and Importing the library

In [ ]:
```
import nltk
nltk.download('gutenberg')
from nltk.corpus import gutenberg
```

```
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Unzipping corpora/gutenberg.zip.
```

Q1:

c. Use the texts in the corpus.

A1.

c. Using file identifers to access the corpus and the text.

In [ ]:
```
gutenberg.fileids()              # File identifers in the Gutenberg Corpus
```

Out[4]:
```
['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']
```

In [ ]:
```
emma = gutenberg.words('austen-emma.txt')      # Contents and Word count of the text 1
print("Contents of the text:", emma)
print("Word Count of the text:", len(emma))
```

```
Contents of the text: ['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', ...]
Word Count of the text: 192427
```

In [ ]:
```
persuasion = gutenberg.words('austen-persuasion.txt')      # Contents and Word count of the text 2
print("Contents of the text:", persuasion)
print("Word Count of the text:", len(persuasion))
```

```
Contents of the text: ['[', 'Persuasion', 'by', 'Jane', 'Austen', '1818', ...]
Word Count of the text: 98171
```

Q1:

d. Create a table displaying relative frequencies with which “modals” (can, could, may, might, will, would, and should) are used in all texts provided in the corpus.

A1.

d. Using NLTK function of Conditional Frequency to display the relative frequencies table for the modals of all the texts in corpus.

In [ ]:
```
freq = nltk.ConditionalFreqDist((id, word)
  for id in gutenberg.fileids()
  for word in gutenberg.words(id))
modals = ['can', 'could', 'may', 'might', 'must', 'will']
freq.tabulate(samples=modals)
```

	can	could	may	might	must	will
austen-emma.txt	270	825	213	322	564	559
austen-persuasion.txt	100	444	87	166	228	162
austen-sense.txt	206	568	169	215	279	354
bible-kjv.txt	213	165	1024	475	131	3807
blake-poems.txt	20	3	5	2	2	3
bryant-stories.txt	75	154	18	23	39	144
burgess-busterbrown.txt	23	56	3	17	14	19
carroll-alice.txt	57	73	11	28	41	24
chesterton-ball.txt	131	117	90	69	81	198
chesterton-brown.txt	126	170	47	71	70	111
chesterton-thursday.txt	117	148	56	71	48	109
edgeworth-parents.txt	340	420	160	127	250	517
melville-moby_dick.txt	220	215	230	183	282	379
milton-paradise.txt	107	62	116	98	66	161
shakespeare-caesar.txt	16	18	35	12	30	129
shakespeare-hamlet.txt	33	26	56	28	53	131
shakespeare-macbeth.txt	21	15	30	5	33	62
whitman-leaves.txt	88	49	85	26	63	261

Q1:

e. For two modals with the largest span of relative frequencies (most used minus least used), select a text which uses it the most and the text that uses it the least. Compare usage in both texts by examining the relative frequencies of those modals in the two texts. Try to explain why those words are used differently in the two texts.

A1.

e. Analyzing the corpus and the two modals for the largest relative frequencies and comparing them to check the usage in the texts.

In [ ]:
```
freq = nltk.ConditionalFreqDist((id, word)
  for id in gutenberg.fileids()
  for word in gutenberg.words(id))
modals1 = ['may', 'will']
freq.tabulate(samples=modals1)
```

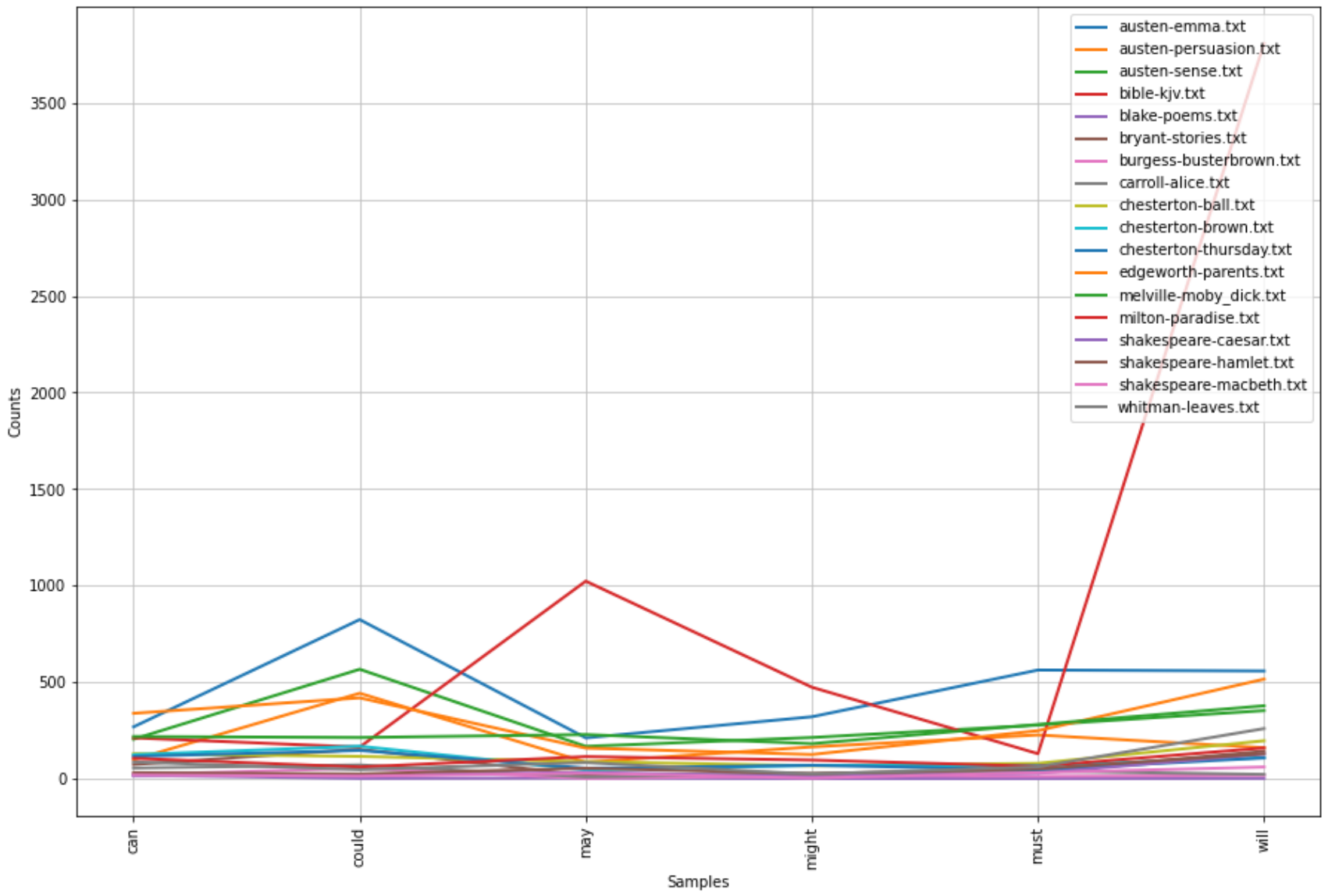
	may	will
austen-emma.txt	213	559
austen-persuasion.txt	87	162
austen-sense.txt	169	354
bible-kjv.txt	1024	3807
blake-poems.txt	5	3
bryant-stories.txt	18	144
burgess-busterbrown.txt	3	19
carroll-alice.txt	11	24
chesterton-ball.txt	90	198
chesterton-brown.txt	47	111
chesterton-thursday.txt	56	109
edgeworth-parents.txt	160	517
melville-moby_dick.txt	230	379
milton-paradise.txt	116	161
shakespeare-caesar.txt	35	129
shakespeare-hamlet.txt	56	131
shakespeare-macbeth.txt	30	62
whitman-leaves.txt	85	261

The text that has the largest span of relative frequencies for the modals is "bible-kjv.txt" which has 3807 frequency count for modal word "will" and 1024 for the word "may" as shown in the above table.

We compare the modal frequency in each text of the corpus to understand which word has the largest frequency and which has the lowest frequency in the corpus.

Comparing all the words of the modal in each text of the corpus

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.figure(figsize=(15, 10))
freq.plot(samples=modals)
```

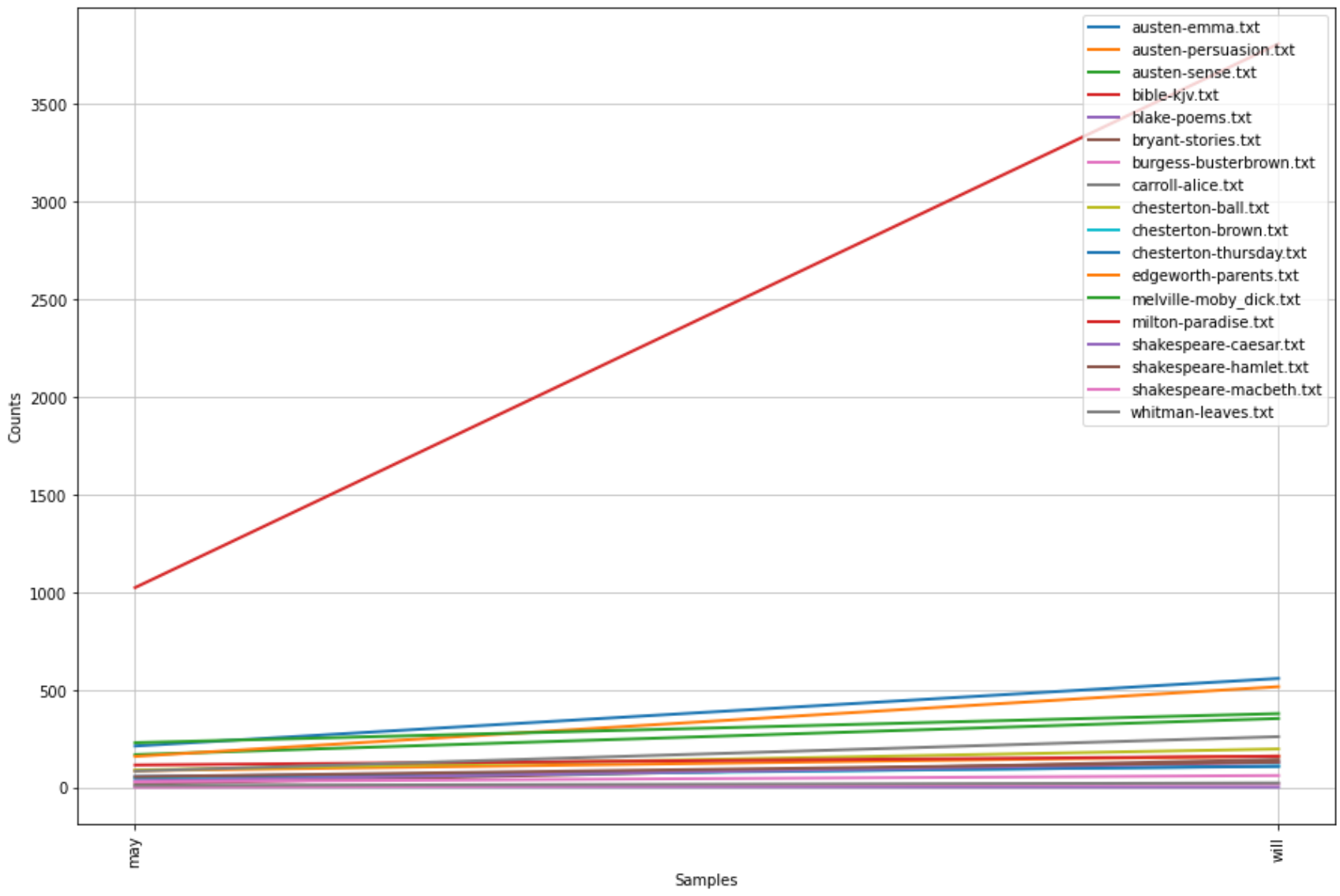


Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f53322cfc50>

Figure 1. Frequency Plot for sample words in the text corpus

Comparing the two modals with largest relative frequency in each text of the corpus

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.figure(figsize=(15, 10))
freq.plot(samples=modals1)
```



Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f533292b5d0>

Figure 2. Frequency Plot for comparing two modals with largest relative frequency in the text corpus

Inaugural Corpus

Q2:

a. In the Inaugural corpus, see below

A2.

a. Downloading corpus and Importing the libraries

In [ ]:
```
nltk.download('inaugural')
from nltk.corpus import inaugural
nltk.download('wordnet')
from nltk.corpus import wordnet
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package inaugural to /root/nltk_data...
[nltk_data]   Unzipping corpora/inaugural.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

Out[11]:
```
True
```

Checking for the Inaugural corpus data

In [ ]:
```
inaugural.fileids()     # File identifers in the Inaugural Corpus
```

Out[12]:
```
['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',
 '2021-Biden.txt']
```

Q2:

b. Chose Kennedy's speech, using e.g., this code

A2.

b. Analysis of the Kennedy Speech

In [ ]:
```
kennedy_words = inaugural.words('1961-Kennedy.txt')
print("Contents of the Kennedy text file are:", kennedy_words)
print("Word count of the text file is:", len(kennedy_words))
```

```
Contents of the Kennedy text file are: ['Vice', 'President', 'Johnson', ',', 'Mr', '.', ...]
Word count of the text file is: 1546
```

Q2:

C. Identify the 10 most frequently used long words (words longer than 7 characters).

A2.

c. List of the 10 most frequently used long words

In [ ]:
```
freq = nltk.FreqDist((word)
    for id in inaugural.fileids()
    for word in inaugural.words(id)
    if len(word) >=7)

top_ten = freq.most_common(10)
top_ten
```

Out[14]:
```
[('Government', 334),
 ('country', 316),
 ('government', 264),
 ('citizens', 241),
 ('America', 240),
 ('Constitution', 200),
 ('nations', 186),
 ('freedom', 180),
 ('American', 171),
 ('national', 138)]
```

Q2:

d. Which one of those 10 words has the largest number of synonyms? Use WordNet as a helper:

A2.

d. Listing the word having the largest number of synonyms

In [ ]:
```
list1 = []
for i in top_ten:
  syns = (wordnet.synsets(i[0]))
  length = len(syns)
  list1.append([i[0], syns, length])

max_list = max(list1, key=lambda sublist: sublist[2])
print(max_list)
print("\nThe word which has the largest number of synonyms is:", max_list[0])
```

```
['national', [Synset('national.n.01'), Synset('national.a.01'), Synset('national.a.02'), Synset('national.a.0
3'), Synset('national.s.04'), Synset('home.s.03'), Synset('national.a.06'), Synset('national.a.07')], 8]

The word which has the largest number of synonyms is: national
```

Q2:

e. List all synonyms for the 10 most frequently used words. Which one of those 10 words has the largest number of hyponyms?

A2.

e. Listing all the synonyms for the frequently used words and printing the word which has the largest number of hyponyms

In [ ]:
```
# List all synonyms for the 10 most frequently used words
print("Listing all the synonyms for the 10 most frequently used words.")
print("\nFrequent Words \t\t Synonyms")
for i in top_ten:
  syns = (wordnet.synsets(i[0]))
  print("\n", i[0],"\t\t",syns)



from itertools import chain
# Finding the Hyponyms of the 10 most frequently used words
list2 = []
for k in top_ten:
  for i,j in enumerate(wordnet.synsets(k[0])):
    #print("Hyponyms of ", k[0], ":", ", ".join(list(chain(*[l.lemma_names() for l in j.hyponyms()]))))
    hyponyms_name = (list(chain(*[l.lemma_names() for l in j.hyponyms()])))
    length2 = len(hyponyms_name)
    list2.append([k[0], length2])

#print(list2)
print("\n\nFinding the word which has the largest number of hyponyms.")
max_list1 = max(list2, key=lambda sublist: sublist[1])
print("\nThe word which has the largest number of hyponyms is", max_list1[0], "with total number of hyponyms of"
```

```
Listing all the synonyms for the 10 most frequently used words.

Frequent Words           Synonyms

 Government              [Synset('government.n.01'), Synset('government.n.02'), Synset('government.n.03'), Syn
set('politics.n.02')]

 country                 [Synset('state.n.04'), Synset('country.n.02'), Synset('nation.n.02'), Synset('countr
y.n.04'), Synset('area.n.01')]

 government              [Synset('government.n.01'), Synset('government.n.02'), Synset('government.n.03'), Syn
set('politics.n.02')]

 citizens                [Synset('citizen.n.01')]

 America                 [Synset('united_states.n.01'), Synset('america.n.02')]

 Constitution            [Synset('fundamental_law.n.01'), Synset('constitution.n.02'), Synset('united_states_c
onstitution.n.01'), Synset('constitution.n.04'), Synset('constitution.n.05')]

 nations                 [Synset('state.n.04'), Synset('nation.n.02'), Synset('nation.n.03'), Synset('nation.
n.04')]

 freedom                 [Synset('freedom.n.01'), Synset('exemption.n.01')]

 American                [Synset('american.n.01'), Synset('american_english.n.01'), Synset('american.n.03'), S
ynset('american.a.01'), Synset('american.a.02')]

 national                [Synset('national.n.01'), Synset('national.a.01'), Synset('national.a.02'), Synset('n
ational.a.03'), Synset('national.s.04'), Synset('home.s.03'), Synset('national.a.06'), Synset('national.a.0
7')]


Finding the word which has the largest number of hyponyms.

The word which has the largest number of hyponyms is American with total number of hyponyms of 99
```

Q2:

f. List all hyponyms of the 10 most frequently used words.

A2.

f. Listing all the hyponyms for the frequently used words

In [ ]:
```
print("Listing all the hyponyms for the 10 most frequently used words.\n")
for k in top_ten:
  for i,j in enumerate(wordnet.synsets(k[0])):
    print("\nHyponyms of", k[0],":",", ".join(list(chain(*[l.lemma_names() for l in j.hyponyms()]))))
```

```
Listing all the hyponyms for the 10 most frequently used words.


Hyponyms of Government : ancien_regime, authoritarian_state, authoritarian_regime, bureaucracy, court, royal
_court, Downing_Street, empire, federal_government, government-in-exile, local_government, military_governme
nt, stratocracy, palace, papacy, pontificate, puppet_government, puppet_state, pupet_regime, state, state_go
vernment, totalitarian_state, totalitation_regime

Hyponyms of Government : legislation, legislating, lawmaking, misgovernment, misrule, trust_busting

Hyponyms of Government :

Hyponyms of Government : geopolitics, realpolitik, practical_politics

Hyponyms of country : ally, city_state, city-state, commonwealth_country, developing_country, Dominion, fore
ign_country, Reich, rogue_state, renegade_state, rogue_nation, sea_power, suzerain, world_power, major_powe
r, great_power, power, superpower

Hyponyms of country : African_country, African_nation, Asian_country, Asian_nation, banana_republic, buffer_
```

Results / Conclusion

From the tasks and analysis implemented for the **Gutenberg corpus** and the **Inaugural corpus**, results obtained help in understanding the texts and data of the corpus. For the gutenberg corpus, the relative frequencies of the specified modals are calculated in order to understand which word has the maximum use of it in a particular text and the least amount of times the word being used in another text of the corpus.

But the analysis was somewhat different for the inaugural corpus as compared to the gutenberg corpus. The task here was to first find the most commonly used words in the corpus using the relative frequency but for the words which are greater than 7 characters in length. The 10 most frequently used words, longer than 7 characters in the inaugural corpus are *Government, country, government, citizens, America, Constitution, nations, freedom American, and national.* Next, the synonyms and hyponyms for these frequently used words were implemented using **WordNet** and the word having largest number of synonyms and hyponyms was displayed, which was *national and American respectively*.

Thus, the respective tasks were performed using Natural Language Processing and NLTK package in order to analyze the textual data of the various corpus.

References

[1] NLTK :: Natural Language Toolkit. (n.d.). https://www.nltk.org/ (https://www.nltk.org/)

[2] GeeksforGeeks. (2019, February 20). NLP | Custom corpus. https://www.geeksforgeeks.org/nlp-custom-corpus/ (https://www.geeksforgeeks.org/nlp-custom-corpus/)

[3] 2. Accessing Text Corpora and Lexical Resources. (n.d.). https://www.nltk.org/book/ch02.html (https://www.nltk.org/book/ch02.html)

[4] US presidential inaugural address texts — data_corpus_inaugural. (n.d.). https://quanteda.io/reference/data_corpus_inaugural.html (https://quanteda.io/reference/data_corpus_inaugural.html)

[5] GeeksforGeeks. (2022b, October 12). NLP | Synsets for a word in WordNet. https://www.geeksforgeeks.org/nlp-synsets-for-a-word-in-wordnet/?ref=lbp (https://www.geeksforgeeks.org/nlp-synsets-for-a-word-in-wordnet/?ref=lbp)

[6] How to get hypernyms and hyponyms for a particular word in nlp -. (n.d.). ProjectPro. https://www.projectpro.io/recipes/get-hypernyms-and-hyponyms-for-particular-word (https://www.projectpro.io/recipes/get-hypernyms-and-hyponyms-for-particular-word)