```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
corpus = []
filenames= []
basepath = '/content/drive/Shared drives/Unstructured Group 5/Data/'
with os.scandir(basepath) as entries:
    for entry in entries:
        if entry.is_file():
            print(entry.name)
            filenames.append(entry.name)
            file_location = basepath + entry.name
            corpus.append(open(file_location,'r').read())
```

```
Polk_1846.txt
Roosevelt_1907.txt
Obama_2016.txt
Taft_1911.txt
Reagan_1986.txt
Roosevelt_1937.txt
Obama_2012.txt
Tyler_1842.txt
Obama_2011.txt
Truman_1953.txt
Truman_1946.txt
Pierce_1854.txt
Wilson_1917.txt
Monroe_1817.txt
Monroe_1819.txt
Roosevelt_1934.txt
Taft_1912.txt
Roosevelt_1903.txt
Monroe_1823.txt
McKinley_1900.txt
Roosevelt_1935.txt
Truman_1949.txt
Wilson_1914.txt
Roosevelt_1906.txt
Roosevelt_1904.txt
Obama_2009.txt
Roosevelt_1941.txt
Roosevelt_1938.txt
Nixon_1971.txt
Roosevelt_1901.txt
Monroe_1820.txt
Pierce_1855.txt
Nixon_1973.txt
Monroe_1818.txt
Taylor_1849.txt
Wilson_1919.txt
```

```
     Washington_1791.txt
     Monroe_1824.txt
     McKinley_1898.txt

     Wilson_1913.txt
     Reagan_1987.txt
     Roosevelt_1936.txt
     Pierce_1853.txt
     Trump_2018.txt
     Roosevelt_1905.txt
     Truman_1947.txt
     Pierce_1856.txt
     Roosevelt_1944.txt
     Monroe_1822.txt
     Tyler_1843.txt
     Washington_1796.txt
     Washington_1792.txt
     Polk_1847.txt
     Truman_1951.txt
     Nixon_1972.txt
     Truman_1948.txt
     Monroe_1821.txt
     Wilson_1920.txt
     Reagan_1988.txt
```

```
#sorting by year
import numpy as np

years = [eval(fname[-8:-4]) for fname in filenames]
year_idx = np.argsort(years)

SOTUcorpus = [corpus[i] for i in year_idx]
SOTUnames = [filenames[i] for i in year_idx]
```

```
!pip install nltk scipy numpy matplotlib scikit-learn
```

```
     Requirement already satisfied: nltk in /usr/local/lib/python3.6/dist-packages (3
     Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (
     Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (
     Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packa
     Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-pac
     Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (fr
     Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-pac
     Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dis
     Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/
     Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/
     Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-pac
```

```
print(len(SOTUcorpus))
```

```
     228
```

Topic analysis

Using Topic Modeling, we will determine the top 7 topics for the State of the Union addresses with the top 10 words being listed for each

```
from sklearn.feature_extraction.text import CountVectorizer


# set max features and whether we want stopwords or note
cvect_corpus = CountVectorizer(stop_words='english', max_features=1000) #only want 10(
X_corpus = cvect_corpus.fit_transform(SOTUcorpus)
vocab_corpus = cvect_corpus.get_feature_names()

from sklearn.decomposition import LatentDirichletAllocation
#Set a seed so that the topic numbers are the same everytime the code is run
import random
random.seed(10)

NUM_TOPICS = 7 #can change to get more topics
lda = LatentDirichletAllocation(n_components=NUM_TOPICS)

lda.fit(X_corpus)

import numpy as np

TOP_N = 10  # change this to see the top N words per topic

topic_norm = lda.components_ / lda.components_.sum(axis=1)[:, np.newaxis]

for idx, topic in enumerate(topic_norm):
    print("Topic id: {}".format(idx))
    #print(topic)
    top_tokens = np.argsort(topic)[::-1] #finding top words in topic
    for i in range(TOP_N):
      print('{}: {}'.format(vocab_corpus[top_tokens[i]], topic[top_tokens[i]]))
    print()
```

```
Topic id: 2
000: 0.022459046219996153
government: 0.019246643928915404
congress: 0.012024708604753798
american: 0.011245729288670066
states: 0.008451052097802236
country: 0.007938428351450063
law: 0.007813906171776101
department: 0.006868024409719941
foreign: 0.006556047472079093
service: 0.006486425311071364

Topic id: 3
government: 0.014310867661636702
law: 0.011129141186427181
great: 0.010567478029229343
people: 0.009863803302329603
```

```
men: 0.009705498099525835
public: 0.008679543505785888
country: 0.008512029799865897
work: 0.008194494744931967
business: 0.008067493643253792
states: 0.008053402266739226

Topic id: 4
world: 0.027293507337320937
war: 0.01962131538561054
peace: 0.01613051931129931
nations: 0.014519853094686501
people: 0.013740390204003246
free: 0.013370689567417847
nation: 0.011657839149498672
great: 0.011280164524313973
freedom: 0.008526650701850656
united: 0.008408249344906455

Topic id: 5
year: 0.019623348736465228
government: 0.018633159320540646
states: 0.015180138080031188
000: 0.014477669476354683
united: 0.0138570984178624
congress: 0.008849311187122332
people: 0.008114526947449945
secretary: 0.007052318596065822
fiscal: 0.00668813706159737
service: 0.006499299553981205

Topic id: 6
states: 0.02418519146597209
government: 0.01827165667697466
united: 0.01616697101180497
congress: 0.01262488142310877
public: 0.010640134693109921
country: 0.009563715754693738
great: 0.008505157579057886
state: 0.00732392823695332
citizens: 0.006870992491931877
war: 0.006742718347851666
```

Next, we will take this and apply it to the first three State of the Unions which all came from George Washington. This is in order to see which topic he wrote under

```
#First 3 from George Washington, need to figure out how to sort by date rather than al

docs_sample = lda.transform(X_corpus[0:3])

for i in range(3):
    print('Document: {}'.format(SOTUcorpus[i][0:300]))
    row = docs_sample[i]
    print(row)
```

```
    print(row)
    top_topics = np.argsort(row)[::-1]
    #print(top_topics[0:3])
    print('top topic: {}'.format(top_topics[0])) #prints first entry in top topics, i
    print("\n")
```

```
Document: Fellow-Citizens of the Senate and House of Representatives:
I embrace with great satisfaction the opportunity which now presents itself of c
[4.96354686e-04 4.96266767e-04 4.96310309e-04 4.96465242e-04
 5.82317034e-02 4.96037706e-04 9.39286862e-01]
top topic: 6


Document: Fellow-Citizens of the Senate and House of Representatives:

"In vain may we expect peace with the Indians on our frontiers so long as a
lawless set of unprincipled wretches can violate the rights of hospitality,
or infringe the most solemn treaties, without receiving the punishment they
so justly m
[2.63653785e-04 2.63578281e-04 2.63707923e-04 2.64000166e-04
 2.63600003e-04 2.63633780e-04 9.98417826e-01]
top topic: 6


Document: Fellow-Citizens of the Senate and House of Representatives:

It is some abatement of the satisfaction with which I meet you on the
present occasion that, in felicitating you on a continuance of the national
prosperity generally, I am not able to add to it information that the
Indian hostilities whic
[2.99493146e-04 2.99441027e-04 2.99520467e-04 2.99648293e-04
 2.99604752e-04 2.99481518e-04 9.98202811e-01]
top topic: 6
```

Now, the last three. One Obama and 2 Trump.

```
# Last 3 SOTU addresses in dataset, Obama's 8th and Trump's 1st and 2nd

docs_sample = lda.transform(X_corpus[-3:])

for i in range(-3,0):
    print('Document: {}'.format(SOTUcorpus[i][0:300]))
    row = docs_sample[i]
    print(row)
    top_topics = np.argsort(row)[::-1]
    #print(top_topics[0:3])
    print('top topic: {}'.format(top_topics[0])) #prints first entry in top topics, i
    print("\n")
```

```
Document: Mr. Speaker, Mr. Vice President, Members of Congress, my fellow
Americans:

Tonight marks the eighth year I've come here to report on the State of
the Union. And for this final one, I'm going to try to make it shorter.
I know some of you are antsy to get back to Iowa.

I also understand that because
[9.67853694e-01 1.11746322e-04 1.11739798e-04 1.11783202e-04
 3.15874265e-02 1.11740657e-04 1.11869396e-04]
top topic: 0


Document: Thank you very much. Mr. Speaker, Mr. Vice President, members of
Congress, the first lady of the United States ...

... and citizens of America, tonight, as we mark the conclusion of our
celebration of Black History Month, we are reminded of our nation's path
toward civil rights and the work that st
[8.45588075e-01 1.15462513e-04 7.62407975e-04 6.33894881e-02
 6.21830696e-02 1.15482224e-04 2.78460146e-02]
top topic: 0


Document: Mr. Speaker, Mr. Vice President, Members of Congress, the First Lady o
the United States, and my fellow Americans:

Less than 1 year has passed since I first stood at this podium, in this
majestic chamber, to speak on behalf of the American People -- and to
address their concerns, their hopes, and
[9.26828500e-01 1.18482437e-04 9.44835950e-03 1.18533613e-04
 1.18523177e-04 6.32490814e-02 1.18519423e-04]
top topic: 0
```

Seeing that the topics were consistent over generational shift, we want to determine how the topics have changed over the generations. In order to do this, we will create an array to store the topic and year, and then create a dictionary to look them up. After this is done, we will create a dataframe in order to visualize all of the decades with their top topics.

```
# Looking at how the topics change with every year
docs_sample = lda.transform(X_corpus)

# Create an array to store the topic / year
Topics = np.zeros(len(docs_sample))
Years = np.zeros(len(docs_sample))

for i in range(len(docs_sample)):
    row = docs_sample[i]
    top_topics = np.argsort(row)[::-1]
    Topics[i] = top_topics[0]
    Years[i] = SOTUnames[i][-8:-4]
```

```
# Create a dictionary to easily lookup topics and years
TopicsByYear = dict(zip(Years, Topics))


# Topics of certain years
year = 1945
print('The topic of {} is: {}'.format(year, TopicsByYear[year]))
```

```
The topic of 1945 is: 4.0
```

```
# What is the most popular topic by Decade?
import pandas as pd
# Create a pandas dataframe to utilize the groupby and agg functions
DecadeData = pd.DataFrame(data = {'Year': Years, 'Topic': Topics})
# Calculate the decade of each topic
DecadeData['Decade'] = DecadeData['Year'] // 10 * 10
# Find the mode topic of each decade
DecadeData.groupby('Decade').agg({'Topic': pd.Series.mode})
```

Decade	Topic
1790.0	6
1800.0	6
1810.0	6
1820.0	6
1860.0	6

Party Affiliation Data Frame

Now we will look to see if the topic changes over parties.

```
df = pd.DataFrame(columns=['file_name', 'year', 'president', 'party', 'text'])
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')

for i in range(len(SOTUnames)):
    components = SOTUnames[i].split('_')
    name = components[0]
    year = components[1].split('.')[0]
    df.loc[i, 'file_name'] = SOTUnames[i]
    df.loc[i,'year'] = year
    df.loc[i,'president'] = name
    df.loc[i, 'text'] = SOTUcorpus[i]
    # df.loc[i, 'tokens'] = nltk.word_tokenize(SOTUcorpus[i])
    # df.loc[i, 'lex_div'] = len(set(nltk.word_tokenize(SOTUcorpus[i]))) / len(nltk.wo
    # df.loc[i, 'len'] = len(nltk.word_tokenize(SOTUcorpus[i]))
    # df.loc[i, 'set'] = len(set(nltk.word_tokenize(SOTUcorpus[i])))

df.year = df.year.astype(int)

# Fix entries where presidents have the same last name
indices = df.query("president =='Roosevelt' & year <= 1909").index
df.loc[indices,'president'] = 'Theodore Roosevelt'

indices = df.query("president == 'Roosevelt'").index
df.loc[indices,'president'] = 'Franklin D. Roosevelt'

indices = df.query("president =='Bush' & year <= 1992").index
df.loc[indices,'president'] = 'George H. W. Bush'

indices = df.query("president == 'Bush'").index
df.loc[indices,'president'] = 'George W. Bush'

indices = df.query("president =='Johnson' & year <= 1869").index
df.loc[indices,'president'] = 'Andrew Johnson'
```

```
indices = df.query("president == 'Johnson'").index
df.loc[indices,'president'] = 'Lyndon B. Johnson'

indices = df.query("president =='Adams' & year <= 1801").index
df.loc[indices,'president'] = 'John Adams'

indices = df.query("president == 'Adams'").index
df.loc[indices,'president'] = 'John Quincy Adams'


indices = df.query("president =='Harrison' & year <= 1841").index
df.loc[indices,'president'] = 'William Henry Harrison'

indices = df.query("president == 'Harrison'").index
df.loc[indices,'president'] = 'Benjamin Harrison'

def pres_to_party(name):
    republican = ['Lincoln', 'Grant', 'Hayes', 'Garfield', 'Arthur',
                  'Benjamin Harrison', 'McKinley', 'Theodore Roosevelt',
                  'Taft', 'Harding', 'Coolidge', 'Hoover', 'Eisenhower',
                  'Nixon', 'Ford', 'Reagan', 'George H. W. Bush',
                  'George W. Bush', 'Trump']
    if name in republican:
        return 'Republican'

    democratic = ['Jackson', 'Buren', 'Polk', 'Pierce',
                  'Buchanan', 'Cleveland', 'Wilson', 'Franklin D. Roosevelt',
                  'Truman', 'Kennedy', 'Lyndon B. Johnson', 'Carter', 'Clinton', 'Obar
    if name in democratic:
        return 'Democratic'

    whig = ['William Henry Harrison', 'Taylor', 'Fillmore']
    if name in whig:
        return 'Whig'

    national_union = ['Andrew Johnson']
    if name in national_union:
        return 'National Union'


    unaffiliated = ['Washington', 'Tyler']
    if name in unaffiliated:
        return 'Unaffiliated'

    federalist = ['John Adams']
    if name in federalist:
        return 'Federalist'

    democratic_republican = ['Jefferson', 'Madison', 'Monroe', 'John Quincy Adams']
    if name in democratic_republican:
        return 'Democratic-Republican'
```

```
        return 'Democratic-Republican'

df.party = df.president.apply(pres_to_party)

# df.set_index('year', inplace=True)
# df.sort_index(inplace=True)
df.sort_values(by=['year'], inplace=True)

df
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

	file_name	year	president	party	text
0	Washington_1790.txt	1790	Washington	Unaffiliated	Fellow-Citizens of the Senate and House of Rep...
1	Washington_1791.txt	1791	Washington	Unaffiliated	Fellow-Citizens of the Senate and House of Rep...
2	Washington_1792.txt	1792	Washington	Unaffiliated	Fellow-Citizens of the Senate and House of Rep...
3	Washington_1793.txt	1793	Washington	Unaffiliated	Fellow-Citizens of the Senate and House of Rep...
4	Washington_1794.txt	1794	Washington	Unaffiliated	Fellow-Citizens of the Senate and House of Rep...
...	...	...	...	...	...
223	Obama_2014.txt	2014	Obama	Democratic	Mr. Speaker, Mr. Vice President, Members of Co...
224	Obama_2015.txt	2015	Obama	Democratic	Mr. Speaker, Mr. Vice President, Members of Co...

```
df['tokens'] = df.apply(lambda row: nltk.word_tokenize(row.text), axis = 1)
df['set_len'] = df.apply(lambda row: len(set(row.tokens)), axis = 1)
df['len'] = df.apply(lambda row: len(row.tokens), axis = 1)
df['lex_div'] = df.apply(lambda row: row.set_len/row.len, axis = 1)

df.head()
```

	file_name	year	president	party	text	tokens	set_len	len	le:
					Fellow-Citizens of the	[Fellow-Citizens,			

Lexical Diversity

This section will show how different the words used are in one speech. This will show how diverse and unique each speech is.

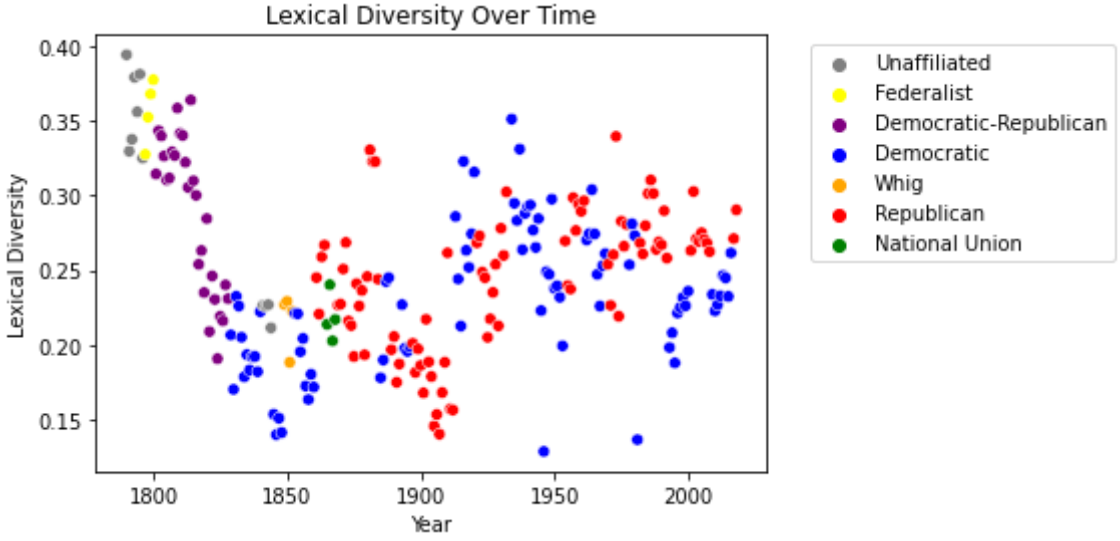
Citizens [Fellow-

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
color_dict = {'Unaffiliated': 'gray', 'Federalist': 'yellow', 'Democratic-Republican':
       'Democratic':'blue', 'Whig':'orange', 'Republican':'red', 'National Union':'gre

g = sns.scatterplot(x=df['year'], y=df['lex_div'], hue=df['party'],
              data=df, palette=color_dict,
                   legend='full')
g.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xlabel('Year')
plt.ylabel('Lexical Diversity')
plt.title('Lexical Diversity Over Time')
```

```
Text(0.5, 1.0, 'Lexical Diversity Over Time')
```



Most Important Terms Based on TF-IDF Scores

After seeing the results from the Lexical Diversity, we want to know the top 5 words of each speech in order to get a sense of patterns throughout Presidents and over time.

```
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
import pandas as pd

def tfidf_vectorizer(corpus):
  cvect = CountVectorizer()
  count_matrix = cvect.fit_transform(corpus)
  tokens = cvect.get_feature_names()

  count_matrix = pd.DataFrame(count_matrix.todense())

  df_vect = count_matrix.astype(bool).sum(axis=0)
  df_vect = np.log(len(corpus) / df_vect)
  print(tokens, np.array(count_matrix * df_vect))
  return tokens, np.array(count_matrix * df_vect)

tokens, tfidf_matrix = tfidf_vectorizer(SOTUcorpus)
print(tfidf_matrix.shape)

idx_to_tokens = {}
tokens_to_idx = {}

for i in range(len(tokens)):
  token = tokens[i]
  tokens_to_idx[token] = i
  idx_to_tokens[i] = token









for i in range(0,len(tfidf_matrix)):
  print("\n", df.file_name[i])
  bookarray = tfidf_matrix[i][:]
  idx = np.argsort(bookarray)
  idx = idx[::-1]
  for i in idx[0:5]:
    print("{}: {}".format(tokens[i], bookarray[i]))
```