Assignment 5: Text Analytics: Bag-of-Words

by Prudvi Kamtam

-------- Final Report --------

Test set accuracy: 95.624%

1. Conduct string manipulation to prepare documents for text analytics

In [ ]:
```
import pandas as pd
import numpy as np

# Load the data into dataframes
user1 = pd.read_csv('AOC_tweets.csv')#['text']
user2 = pd.read_csv('realDonaldTrump_tweets.csv')#['text']
```

In [ ]:
```
import re
import nltk
import ssl
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# This fails for some reason
# nltk.download('stopwords')
# tried
# https://stackoverflow.com/a/50406704

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.
xml
```

Out[ ]:
```
True
```

In [ ]:
```
stop_words = set(stopwords.words('english'))
def preprocess(text):
    # Convert text to lowercase
    text = text.lower()
    # Remove URLs
    text = re.sub(r'http\S+', '', text)
    # Remove mentions
    text = re.sub(r'@\S+', '', text)
    # Remove special characters and digits
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\d+', '', text)
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stop words
    tokens = [token for token in tokens if token not in stop_words]
    # Join the tokens back into a string
    text = ' '.join(tokens)
    return text

# Apply the preprocessing function to the tweets
user1['clean_text'] = user1['text'].apply(preprocess)
user2['clean_text'] = user2['text'].apply(preprocess)
```

In [ ]:
```
user1['clean_text'].head()
```

Out[ ]:
```
0          want learn violence interruption program read
1     violence interruption works keep us safe job b...
2     bronx made public hospital response center vio...
3     reduce gun violence without expanding mass inc...
4              someone call lucia seamstress fix believe
Name: clean_text, dtype: object
```

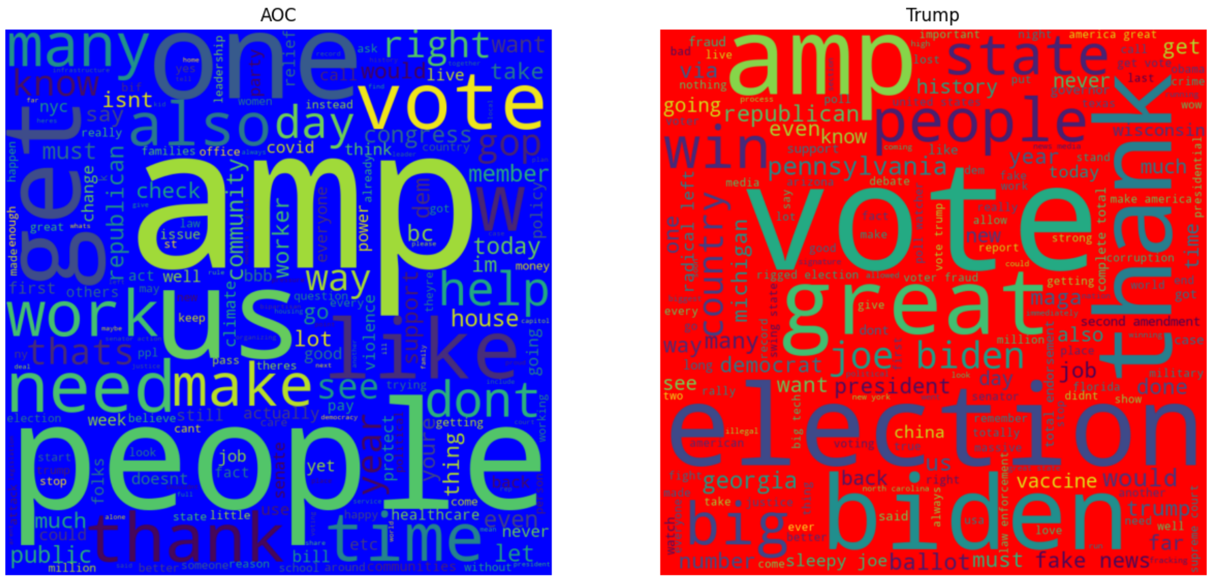
2. Convert the data to matrix format (document-term matrix)

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()

# Fit and transform the preprocessed text
user1_vectorized = vectorizer.fit_transform(user1['clean_text'])
user2_vectorized = vectorizer.fit_transform(user2['clean_text'])
```

3. Visualizations to compare/contrast the posts from your two candidates

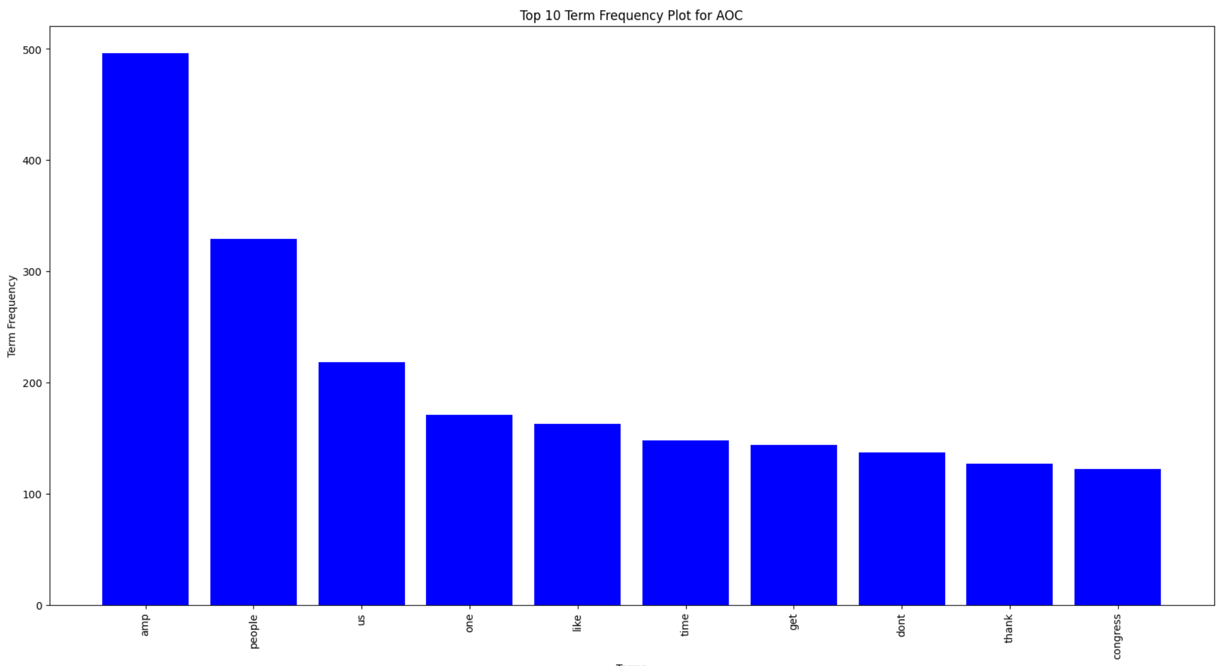
In [ ]:
```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Create word clouds for each user
wordcloud1 = WordCloud(width = 800, height = 800,
                background_color ='blue',
                stopwords = stop_words,
                min_font_size = 10).generate(' '.join(user1['clean_text']))

wordcloud2 = WordCloud(width = 800, height = 800,
                background_color ='red',
                stopwords = stop_words,
                min_font_size = 10).generate(' '.join(user2['clean_text']))

# Plot the word clouds
fig, axs = plt.subplots(1, 2, figsize=(15, 15))
axs[0].imshow(wordcloud1, interpolation='bilinear')
```

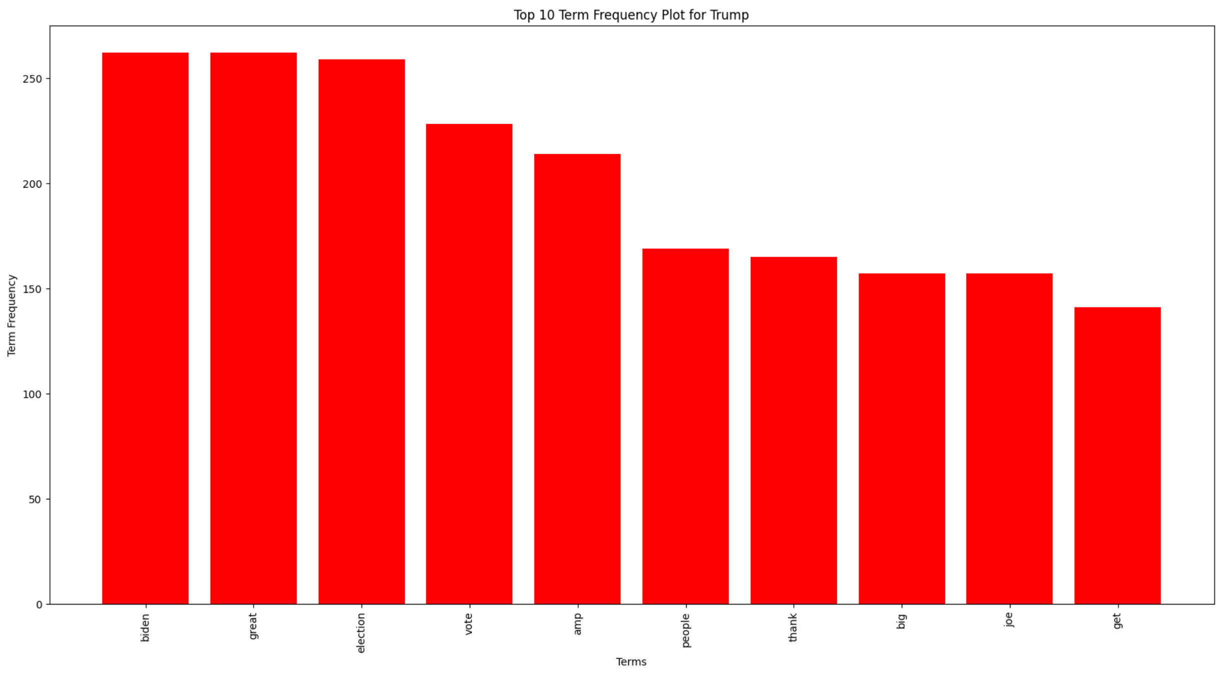
```
axs[0].set_title('AOC')
axs[0].axis("off")
axs[1].imshow(wordcloud2, interpolation='bilinear')
axs[1].set_title('Trump')
axs[1].axis("off")
plt.show()
```



```
In [ ]: from collections import Counter

        # Initialize the CountVectorizer object
        vectorizer = CountVectorizer()

        # Fit and transform the preprocessed text for user 1
        user1_vectorized = vectorizer.fit_transform(user1['clean_text'])
        user1_term_freq = np.asarray(user1_vectorized.sum(axis=0))[0]
        user1_term_freq_dict = Counter(dict(zip(vectorizer.get_feature_names_out(),

        # Get the top 10 terms
        user1_top_terms = dict(user1_term_freq_dict.most_common(10))

        # Create a bar plot of the top 10 terms for politician 1
        plt.figure(figsize=(20, 10))
        plt.bar(user1_top_terms.keys(), user1_top_terms.values(), color='blue')
        plt.xticks(rotation=90)
        plt.title('Top 10 Term Frequency Plot for AOC')
        plt.xlabel('Terms')
        plt.ylabel('Term Frequency')
        plt.show()


        # Fit and transform the preprocessed text for user 2
        user2_vectorized = vectorizer.fit_transform(user2['clean_text'])
        user2_term_freq = np.asarray(user2_vectorized.sum(axis=0))[0]
        user2_term_freq_dict = Counter(dict(zip(vectorizer.get_feature_names_out(),

        # Get the top 10 terms
        user2_top_terms = dict(user2_term_freq_dict.most_common(10))
```

```
# Create a bar plot of the top 10 terms for politician 2
plt.figure(figsize=(20, 10))
plt.bar(user2_top_terms.keys(), user2_top_terms.values(), color='red')
plt.xticks(rotation=90)
plt.title('Top 10 Term Frequency Plot for Trump')
plt.xlabel('Terms')
plt.ylabel('Term Frequency')
plt.show()
```





4. Partition the documents into train-test subsets

In [ ]:
```
print(user1_vectorized[:, :5])
```

```
  (160, 4)      1
  (162, 2)      1
  (404, 3)      1
  (1014, 1)     1
  (1297, 0)     1
  (1322, 1)     1
  (1855, 1)     1
```

In [ ]:

```
from sklearn.model_selection import train_test_split

# Combine the two dataframes
politicians = pd.concat([user1, user2], axis=0)

# Create the target variable
politicians['target'] = np.where(politicians.index.isin(user1.index), 'user1

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(politicians['clean_text'
```

In [ ]:

```
politicians.head()
```

Out[ ]:

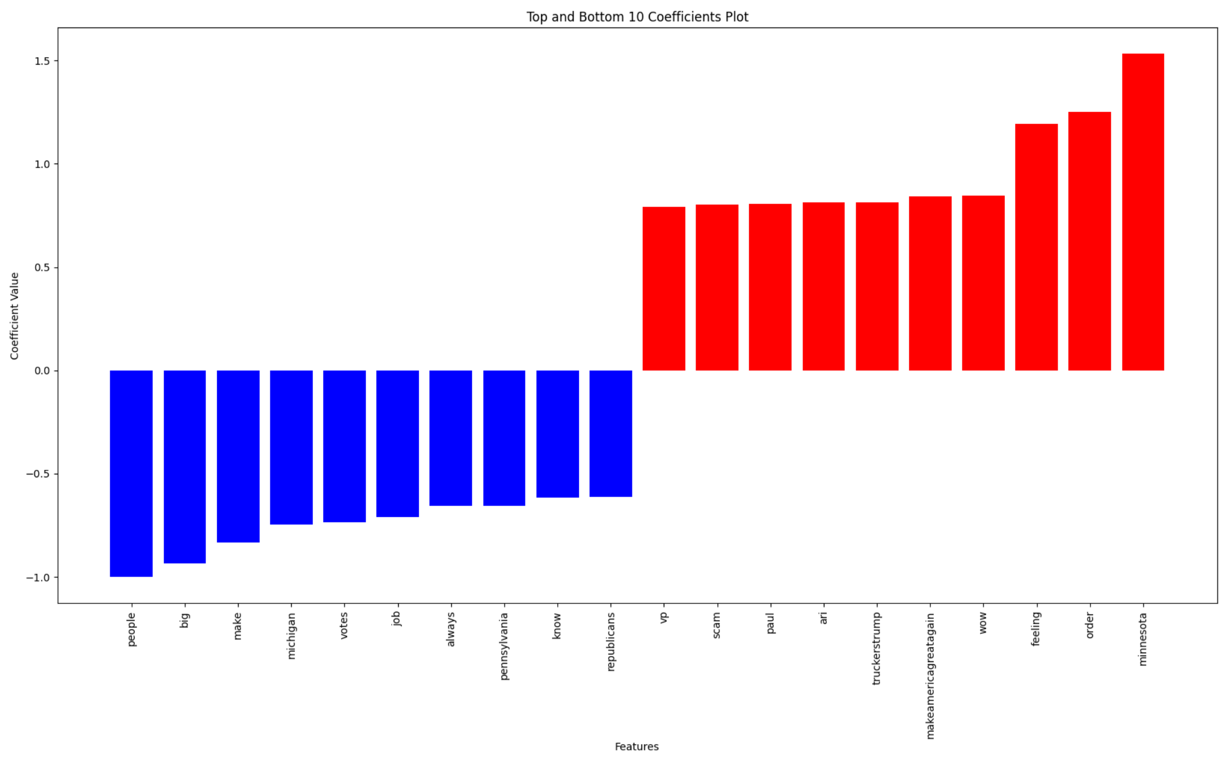
	id	created_at	favorite_count	retweet_count	text
0	1648362209448189953	2023-04-18 16:25:50+00:00	954	117	Want to learn more about our violence interrup...
1	1648360873935659010	2023-04-18 16:20:32+00:00	1518	106	Violence interruption works to keep us safe. O...
2	1648357478336286720	2023-04-18 16:07:02+00:00	7615	1024	In the Bronx, when we made our public hospital...
3	1648122147125047297	2023-04-18 00:31:55+00:00	5452	644	We can reduce gun violence without expanding m...
4	1647756915114377217	2023-04-17 00:20:37+00:00	99106	13964	Someone call Lucia the seamstress to fix this....

5. Build a logistic regression model with the terms as your predictors and politician as the target variable

```
In [ ]: from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.linear_model import LogisticRegression
        from sklearn.pipeline import Pipeline

        vectorizer = CountVectorizer()

        logreg = LogisticRegression(random_state=42)

        # Create the pipeline object
        pipeline = Pipeline([('vectorizer', vectorizer), ('logreg', logreg)])
        pipeline.fit(X_train, y_train)

        y_pred = pipeline.predict(X_test)
```

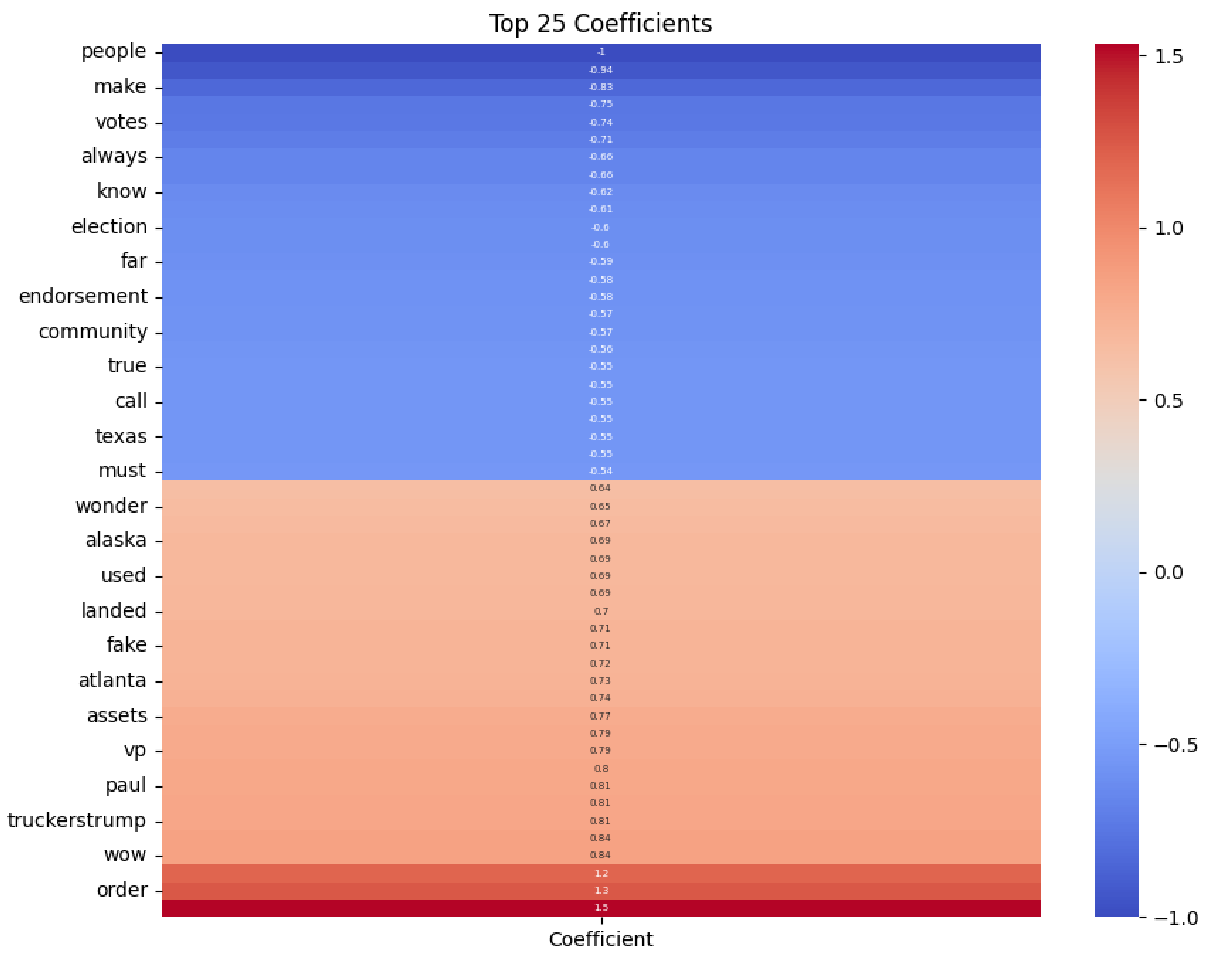
6. Visualize the test statistic for each term's coefficient

```
In [ ]: # Get the coefficients of the logistic regression model
        coef = logreg.coef_[0]

        feature_names = vectorizer.get_feature_names_out()

        df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coef})

        # sorting the dataframe by coefficient value
        df = df.sort_values(by='Coefficient', ascending=False)

        plt.figure(figsize=(20, 10))

        top = 10

        sorted_list = [t for t in sorted(zip(list(df['Coefficient']), list(df['Featu

        feat = [t[1] for t in sorted_list]
        coeff = [t[0] for t in sorted_list]

        plt.bar(feat[:top], coeff[:top], color='blue') # AOC
        plt.bar(feat[-top:], coeff[-top:], color='red') # Trump
        plt.xticks(rotation=90)
        plt.title(f'Top and Bottom {top} Coefficients Plot')
        plt.xlabel('Features')
        plt.ylabel('Coefficient Value')
        plt.show()
```



In [ ]:
```
# sorted_list = [t for t in sorted(zip(list(df['Coefficient']), list(df['Fea
# # sorted_list[:10], sorted_list[-10:]
# feat = [t[1] for t in sorted_list]
# coeff = [t[0] for t in sorted_list]

# feat[:10], coeff[-10:]
```

In [ ]:
```
import seaborn as sns

top = 25
sorted_list = [t for t in sorted(zip(list(df['Coefficient']), list(df['Featu

feat = [t[1] for t in sorted_list]
coeff = [t[0] for t in sorted_list]

feat = feat[:top] + feat[-top:]
coeff = coeff[:top] + coeff[-top:]

coef_df = pd.DataFrame.from_dict(dict(zip(feat, coeff)), orient='index', col

# Plot the heatmap
plt.figure(figsize=(10,8))
sns.heatmap(coef_df, cmap='coolwarm', annot=True, annot_kws={'fontsize':5})
plt.title(f'Top {top} Coefficients')
plt.show()
```



7. Test set accuracy

```
In [ ]: from sklearn.metrics import accuracy_score

        # calculate the accuracy of the model on the test set
        accuracy = accuracy_score(y_test, y_pred)

        print('----- Final Report -----')
        print(f'Test set accuracy: {accuracy*100:.3f}%')
```

```
----- Final Report -----
Test set accuracy: 95.624%
```

```
In [ ]:
```