I have a Dream
Text Analysis

Heejae Roh

Northeastern University

ALY 6040: Data Mining

Ahmadi Behzad

May 14, 2023

Introduction

In this analysis, I will be focusing on the text mining process, which is a crucial aspect of data mining. As technologies like ChatGPT and Bard emerge, designed to generate text-based content, the importance of text mining is becoming increasingly significant. Recently, Google introduced a new version of Bard at their annual developer conference held in Mountain View, California, which was quite surprising.

Bard is a chatbot that utilizes a large language model (LLM) developed by Google AI. It is trained on a massive dataset of text and code and can perform tasks such as generating text, translating languages, creating different types of creative content, and answering questions informatively (Bard, 2023).

For this project, I have chosen to use the "I Have a Dream" speech delivered by Martin Luther King Jr. on August 28, 1963, in Washington, D.C., USA. This speech was given from the balcony of the Lincoln Memorial, overlooking the Washington Monument and is still regarded as one of the greatest speeches in human history. Famous sentences from this speech are still published in textbooks or quoted by celebrities in their speeches. To challenge myself and become more proficient in Python coding, I will be using Python instead of R.

First look of text dataset

Tokenization

- In Python tokenization basically refers to splitting up a larger body of text into smaller lines, words or even creating words for a non-English language (tutorialspoint, 2023).
- NLTK library is used for word_tokenize function.

'I Have a Dream' text contains 688 words before removing stop-words.

Whole text	And so even though we face the difficulties of today and tomorrow, I still have a dream. It is a dream deeply rooted in the American dream. I have a dream that one day this nation will rise up and live out the true meaning of its creed: (…) we are free at last!
Tokens	['And', 'so', 'even', 'though', 'we', 'face', 'the', 'difficulties', 'of', 'today', 'and', 'tomorrow', ',', 'I', (…) 'we', 'are', 'free', 'at', 'last', '!']

This dataset includes only text:

- Frequency of whole text: ({',': 39, 'of': 34, 'the': 31, '.': 23, 'and': 21, 'be': 20, 'will': 17, 'to': 16, 'a': 15, 'freedom': 13, ...})
- Containing lots of stop-words which can be removed with text pre-processing
- Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence (tutorialspoint, 2023).

Text pre-processing 1: removing stop-words

Removing stop-words:

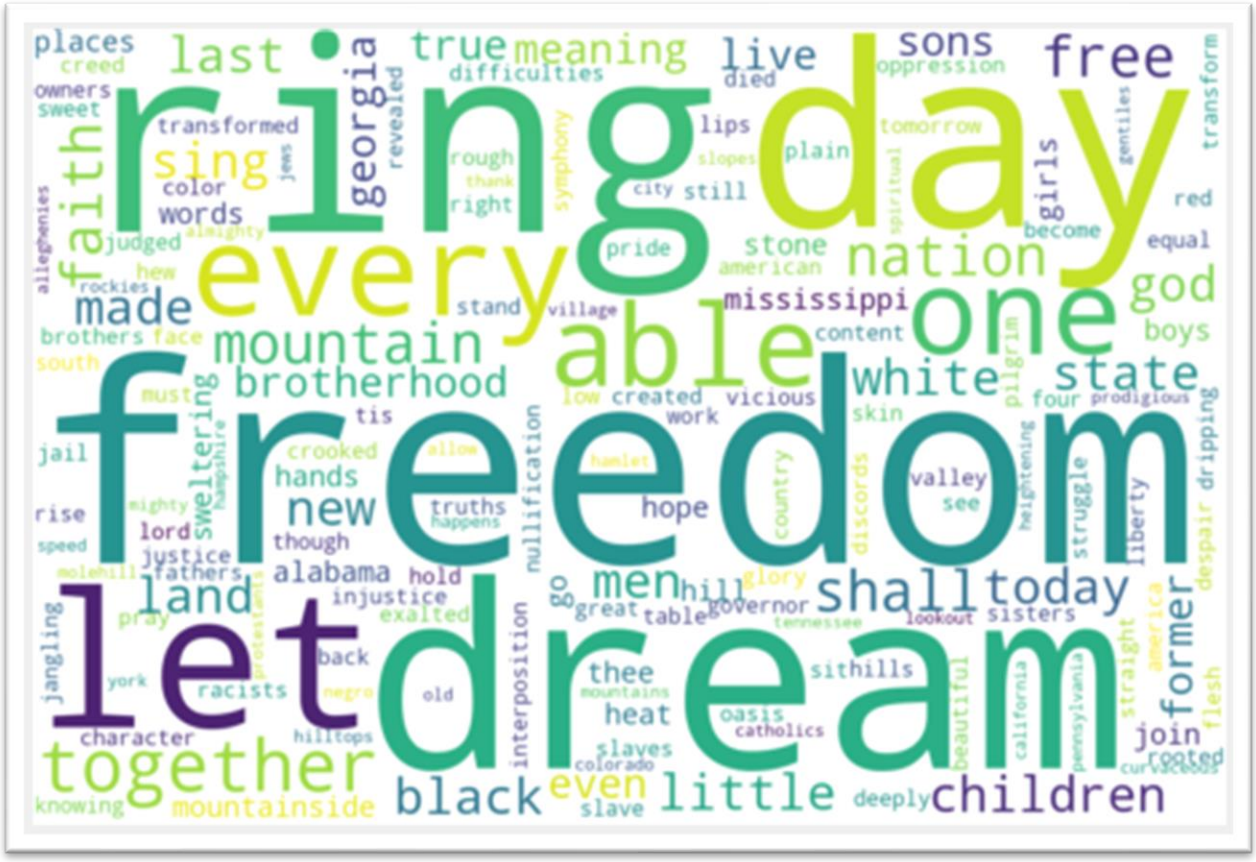
- With NLTK library in python, using stopwords.words('english'), I check the stop-words in whole text from martin Luther King Jr.
- After removing stop-words, total tokenized text reduces to 310 from 688. It means 54.9% of tokenized text is stop-words.
- Stop words are words that are so common they are basically ignored by typical tokenizers. By default, NLTK (Natural Language Toolkit) includes a list of 40 stop words, including: "a", "an", "the", "of", "in", etc. The stopwords in nltk are the most common words in data (Python Tutorials, 2023).

Stopwords in NLTK library ('english')	{"shouldn't", 'other', 'my', 'not', "needn't", 'as', 'it', 'should', 'needn', 'of', 'his', 'am', 'hers', 'm', 'their', 'y', 'above', 'him', 'them', 'having', 're', 'shouldn', 'from', 'your', "she's", "haven't", 'i', 'they', 'or', 'below', 've', 'until', "couldn't", 'during', 'again', 'weren', 'he', 'themselves', "wouldn't", 'after', "should've", 'for', "don't", 'yourself', 'few', 'shan', 'me', 'under', 'her', 'further', 'ma', 'wouldn', 'will', 'doing', 'and', 'but', 'its', 'doesn', 'wasn', 'which', 'herself', 'don', 'hasn', 'isn', 'with', 'in', 'mustn', 'up', 'at', 'just', 'we', 'won', 'ours', 'can', 'against', "won't", 'o', "aren't", 'very', "mightn't", 'how', 'now', 'do', 'if', 'than', 'd', 'myself', 'you', 'have', 'why', 'while', 'no', "you'd", 'our', 'ourselves', 'couldn', 'mightn', 'so', 'she', 'didn', "isn't", "wasn't", 'through', 'who', 'then', 'same', "that'll", 'did', 'once', 'yours', 't', 'out', 'is', 'most', 'all', 'to', 'each', 'too', 'that', 'ain', 'between', 'itself', 'this', 'about', 'theirs', 'what', 'these', 'some', 'the', 'before', "you're", 'more', 'was', 'there', 'down', "hadn't", 'into', "shan't", 'by', 'own', 'those', 'has', 'had', 'yourselves', 'himself', 'been', 's', 'off', 'whom', "didn't", 'over', "you've", 'be', 'such', "it's", 'both', "you'll", 'aren', "weren't", 'were', 'here', 'any', 'when', 'are', 'an', 'hadn', 'nor', "hasn't", 'only', 'does', 'being', 'because', 'on', 'haven', "doesn't", "mustn't", 'where', 'll', 'a'}

After text pre-processing (removing stop-words):

- Frequency after text pre-processing: ({'freedom': 13, 'ring': 12, 'dream': 11, 'day': 11, 'let': 11, 'every': 9, 'one': 8, 'able': 8, 'together': 7, 'nation': 4, ...})
- Stop-words (above table) are removed with NLTK library. NLTK is a standard python library with prebuilt functions and utilities for ease of use and implementation. It is one of the most used libraries for natural language processing and computational linguistics (Great Learning Team, 2022).

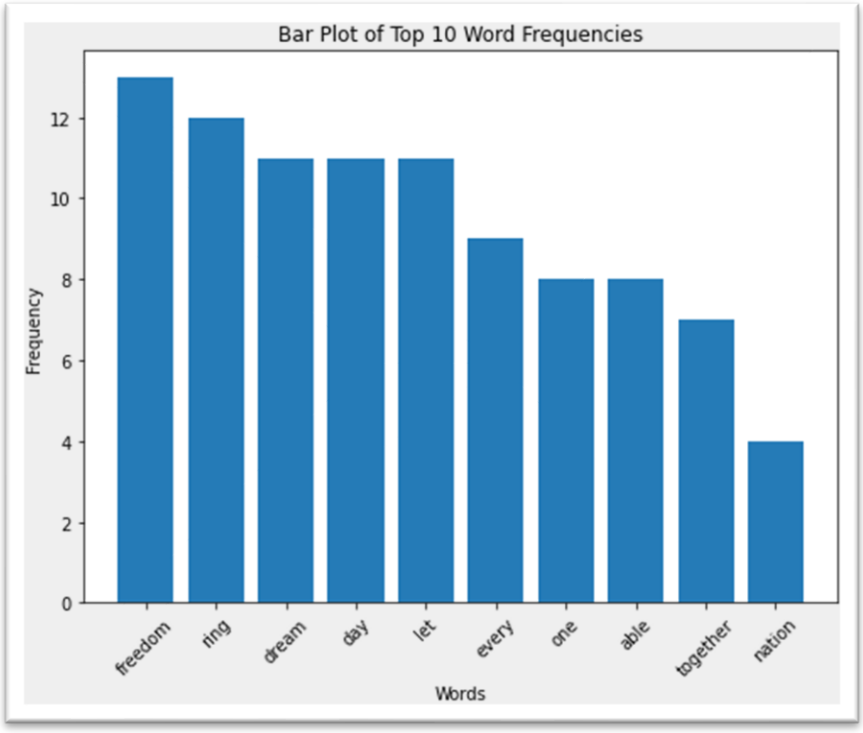
Word cloud after text-preprocessing (removing stop-words)



Interpretation:

- The size of each word in the word cloud is proportional to its frequency. ‘Freedom’ is most frequently appeared word. The count of ‘freedom’ is 13. The other words like ‘ring’, ‘dream’, ‘day’, ‘let’, and ‘every’ follows and the count of each word is 12, 11, 11, 11, 9 respectively.
- It is shown on the word cloud above, Freedom is centralized and the biggest word among them. ‘ring’ and ‘dream’ are the second biggest words on the word cloud. Martin Luther King Jr. used the metaphorical phrase “let freedom ring” several times to refer to the idea of freedom spreading throughout the United States like the ringing of a bell (Laurel, 2020).
- We can compare word cloud after text-preprocessing with word cloud before text-preprocessing (in Appendix). There are ‘,’, ‘of’, ‘the’, ‘.’, ‘and’ and ‘be’, over 20 times each. Makes it hard to understand meaning of whole text, because there are a lot of meaningless words. Therefore, it’s important to remove stop-words before text mining.

Bar plot of Top 10 word frequencies after removing stop-words



Text pre-processing 2: Lemmatization

Lemmatization in NLP:

- Lemmatization is the process of grouping together different inflected forms of the same word. It's used in computational linguistics, natural language processing (NLP) and chatbots (Alexander, n.d).
- Lemmatization and stop word removal are both potentially useful steps in preprocessing text, but they are not necessarily necessary. In order to determine whether either or both such steps should be taken, it is important to consider the nature of the problem. TF-IDF (Term Frequency — Inverse Document Frequency) and LDA (Latent Dirichlet allocation) therefore both benefit in general from lemmatization (Modelop, 2019).

After text pre-processing (removing stop-words):

- Frequency after text pre-processing: ({'freedom': 13, 'ring': 12, 'dream': 11, 'day': 11, 'let': 11, 'every': 9, 'one': 8, 'able': 8, 'together': 7, 'mountain': 5, ...})

Interpretation:

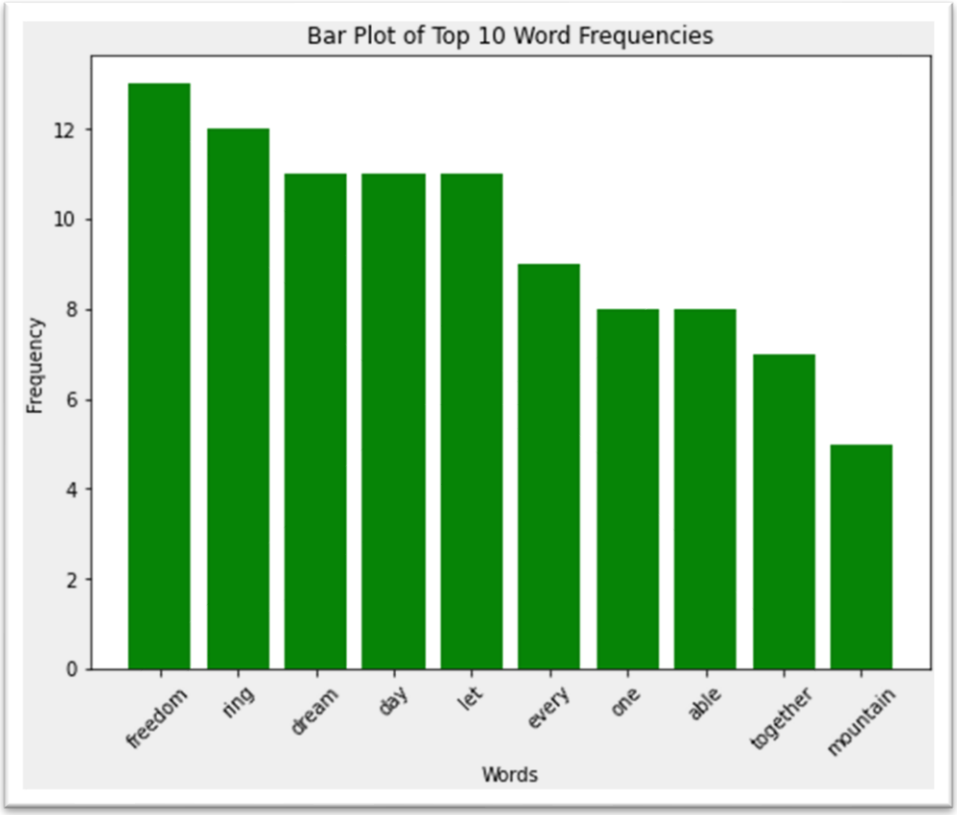
- Lemmatization is the process of reducing words to their base or root form, so that different forms of the same word (e.g. "walk", "walks", "walked") are treated as the same word. This is important because it reduces the complexity of the vocabulary, and allows for more accurate analysis of the text. However, this process may change the meaning of the words in the text.
- Stop-word removal, on the other hand, is the process of removing commonly used words in a language, such as "the", "and", "a", etc. from text data. These words do not usually carry much meaning on their own, and removing them can reduce noise in the data and make it easier to identify the important words in a text. However, this process does not change the words themselves or their meaning.
- Therefore, while both stop-word removal and lemmatization are useful techniques for processing natural language data, they address different aspects of the text and can result in different outputs.

Word cloud after text-preprocessing (removing stop-words)

Difference between Stop-words removing and Lemmatization:

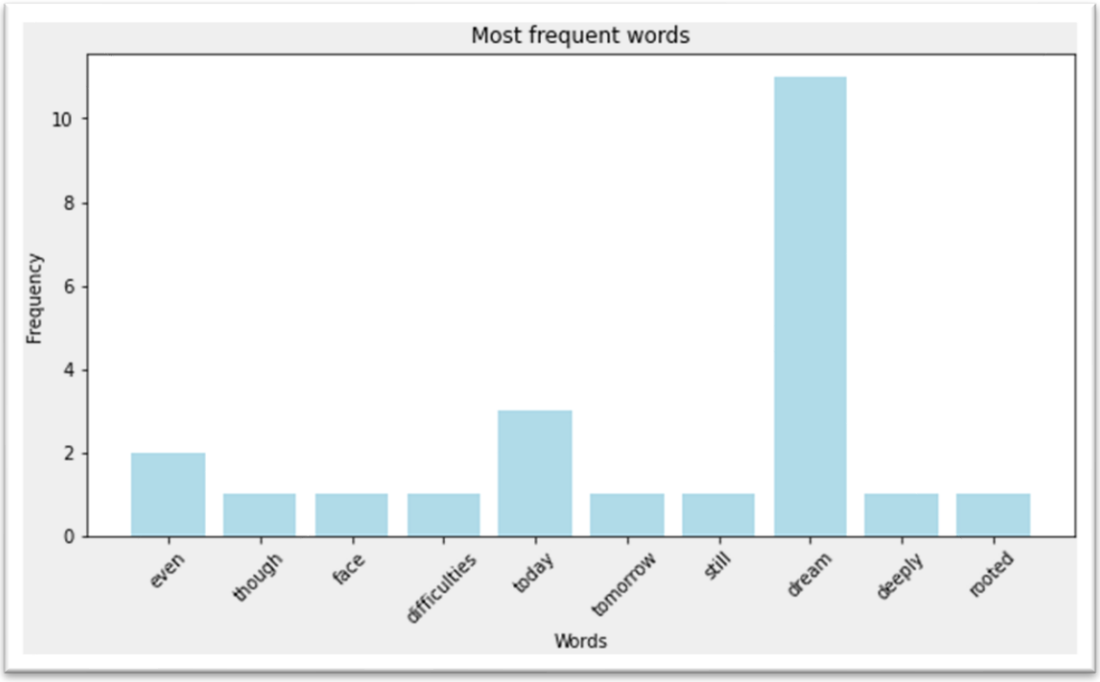
- After applying both stop-word removal and lemmatization, we can observe a difference in the distribution of the bar plot. Specifically, we can see a difference in the 10th most common word between the two approaches, which is "nation" and "mountain".
- This difference is due to the fact that the word "mountainside" in the original text was lemmatized to the root word "mountain", which increased the frequency count for the word "mountain". On the other hand, the word "nation" continues to appear frequently in the text and is not combined with other words, which explains why it remains the 10th most common word even after stop-word removal.
- Overall, this highlights the importance of carefully selecting and applying text preprocessing techniques based on the specific context and goals of the analysis.

Bar plot of Top 10 word frequencies after removing stop-words



Associated Word with ‘freedom’

Bar plot of associated with ‘freedom’



Interpretation:

- These are the result of creating a list of associated terms that contain the word 'freedom', then creating a frequency distribution of the associated terms.
- Generating a bar plot of the top 10 most frequent words in the text. 'dream' is the first most frequent word among them, with 11 counts. 'today' and 'even' follow.
- Through this, we can see that Martin Luther King Jr. used the word freedom a lot along with the words ‘dream’ and ‘today’, and that the above words were also used together with ‘freedom’ as Top 10.

Recommendations

While Text Analysis:

- Before beginning any text analysis project, it is essential to understand the type and properties of the text in question. This allows for the design of a tailored processing approach that best fits the specific needs of the data.
- Preprocessing the text is a critical step in text analysis. It involves the removal of noise, such as punctuations like . , ! $( ) * % @/ , URLs, and stop words, as well as lowercasing, tokenization, stemming, and lemmatization (Deepanshi, 2021). These actions can significantly improve the accuracy of the analysis.
- Representing text as numerical vectors is necessary for performing machine learning on

text data. Several techniques for vectorization exist, including bag-of-words, TF-IDF, and word embeddings. The selection of the best technique depends on the intended use case and specific requirements.

- Visualization is a powerful tool for analyzing text data. It allows for the assessment of pre-processing and final processing of the text, and ensures that the analysis aligns with the type and properties of the text.
- When performing machine learning on text data, it is crucial to evaluate models using metrics such as accuracy, precision, recall, and F1 score. Cross-validation and grid search are additional techniques that can be used to fine-tune model parameters and optimize performance. It is important to carefully consider the ethical implications of text analysis, such as privacy concerns, bias, and discrimination, and take steps to mitigate any issues.

Conclusion

This module provided me with an understanding of the preprocessing steps necessary for text analysis. Using Martin Luther King Jr.'s historic speech as a case study, I explored the differences between lemmatization and stop-word removal. Additionally, I created a word cloud including stop-words in the appendix and compared it to the word cloud generated after their removal. Overall, this exercise demonstrated the importance of proper text preprocessing for accurate analysis and interpretation of textual data.

During my analysis of text data, I gained valuable insights into effective text analysis techniques. Specifically, I recognized the critical importance of conducting an initial analysis of the text and designing a comprehensive processing approach tailored to the specific properties and characteristics of the data.

Furthermore, I re-emphasized the significance of visualizing the text analysis process as an essential tool for evaluating the accuracy and effectiveness of the analysis. Careful consideration and utilization of visualization techniques can facilitate deeper understanding and insight into the text data and ensure that the analysis aligns with the intended objectives.

REFERENCE

sthda. (n.d.). I have a dream. Retrieved from http://www.sthda.com/sthda/RDoc/example-files/martin-luther-king-i-have-a-dream-speech.txt

Bard. (2023, May 12). What is bard?. Retrieved from https://bard.google.com/

tutorialspoint. (2023). Python - remove stopwords. Retrieved from https://www.tutorialspoint.com/python_text_processing/python_remove_stopwords.htm#:~:text=Stopwords%20are%20the%20English%20words,it%20to%20our%20python%20environment.

Python Tutorials. (2023). NLTK stop words. Retrieved from https://pythonspot.com/nltk-stop-words/#:~:text=Stop%20words%20are%20words%20that,most%20common%20words%20in%20data.

Great Learning Team. (2022, October 24). Natural language toolkit (NLTK) tutorial with python. Retrieved from https://www.mygreatlearning.com/blog/nltk-tutorial-with-python/#:~:text=NLTK%20is%20a%20standard%20python,language%20processing%20and%20computational%20linguistics.

Laurel. (2020, July 30). What is the meaning of "let freedom ring"?. Retrieved from https://ell.stackexchange.com/questions/255306/what-is-the-meaning-of-let-freedom-ring

tutorialspoint. (2023). Python - tokenization. Retrieved from https://www.tutorialspoint.com/python_text_processing/python_tokenization.htm#:~:text=In%20Python%20tokenization%20basically%20refers,for%20a%20non%2DEnglish%20language.

Alexander S. Gillis. (n.d). lemmatization. TechTarget. Retrieved from https://www.techtarget.com/searchenterpriseai/definition/lemmatization

Modelop. (2019, March 21). When (not) to lemmatize or remove stop words in text preprocessing. Retrieved from https://www.modelop.com/blog/when-not-to-lemmatize-or-remove-stop-words-in-text-preprocessing/

geeksforgeeks. (2023). Text analysis in Python 3. Retrieved from https://www.geeksforgeeks.org/text-analysis-in-python-3/

Deepanshi. (2021, June 25). Text preprocessing in NLP with python codes. Retrieved from https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/

Appendix

Word cloud before text-preprocessing

Interpretation:

- The English language contains a significant number of auxiliary words, such as "of," "be," and "the," which often do not contribute to the central meaning of a sentence or passage. The presence of these words can potentially hinder the accuracy of text analysis by obscuring the intended message.
- Consequently, it is imperative to eliminate these extraneous words in the text analysis process to extract the underlying essence of the content. Doing so allows for a more focused and precise analysis, ultimately leading to more accurate insights and conclusions.

Python-Code

```
# In[205]:
get_ipython().system('pip install wordcloud')

# In[206]:
get_ipython().system('pip install nltk')

# In[207]:
get_ipython().system('pip install matplotlib')

# In[208]:
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import urllib.request

# In[209]:
import nltk
nltk.download('stopwords')

# In[210]:
url = "http://www.sthda.com/sthda/RDoc/example-files/martin-luther-king-i-
have-a-dream-speech.txt"
response = urllib.request.urlopen(url)
text = response.read().decode('utf-8')

# In[211]:
print(text)

# In[212]:
print(text.find('freedom'))

# In[213]:
tokens = word_tokenize(text)

# In[214]:
print(tokens)

# In[215]:
fdist2 = FreqDist(tokens)
```

```
# In[216]:
fdist2.N()

# In[217]:
fdist2

# In[218]:
stop_words = set(stopwords.words('english'))
words = [word.lower() for word in tokens if word.isalpha() and word.lower()
not in stop_words]

# In[219]:
print(words)

# In[220]:
fdist = FreqDist(words)
d = dict(fdist)

# In[221]:
fdist.N()

# In[222]:
fdist

# In[223]:
wordcloud = WordCloud(width=1200, height=800, background_color='white',
min_font_size=10).generate_from_frequencies(d)
plt.figure(figsize=[8,8])
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

# In[224]:
top_words = fdist.most_common(10)
words, frequencies = zip(*top_words)

# In[225]:
plt.figure(figsize=(8, 6))
plt.bar(words, frequencies)
plt.xticks(rotation=45)
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Bar Plot of Top 10 Word Frequencies')
plt.show()
```

```
# In[226]:
assoc_terms = [word for word in words if 'freedom' in word]
assoc_freq = FreqDist(assoc_terms)
assoc_freq = dict(assoc_freq)
assoc_terms = [word for word, freq in assoc_freq.items() if freq >=
0.3*len(assoc_terms)]

# In[227]:
plt.figure(figsize=(10,5))
plt.bar(list(d.keys())[:10], list(d.values())[:10], color='lightblue')
plt.xticks(rotation=45)
plt.title('Most frequent words')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.show()

# In[228]:
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
print(stop_words)

# In[229]:
fdist2


# In[230]:
e = dict(fdist2)

# In[231]:
wordcloud = WordCloud(width=1200, height=800, background_color='white',
min_font_size=10).generate_from_frequencies(e)
plt.figure(figsize=[8,8])
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()


# In[232]:
from nltk.stem import WordNetLemmatizer

# In[233]:
lemmatizer = WordNetLemmatizer()

# In[234]:
```

```
words3 = [lemmatizer.lemmatize(word.lower()) for word in tokens if
word.isalpha() and word.lower() not in stopwords.words('english')]

# In[235]:
fdist3 = FreqDist(words3)

# In[236]:
f = dict(fdist3)

# In[237]:
fdist3

# In[238]:
wordcloud = WordCloud(width=1200, height=800, background_color='white',
min_font_size=10).generate_from_frequencies(f)

# In[239]:
plt.figure(figsize=[8,8])
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

# In[240]:
top_words = fdist3.most_common(10)
words3, frequencies3 = zip(*top_words)

# In[241]:
plt.figure(figsize=(8, 6))
plt.bar(words3, frequencies3, color='green')
plt.xticks(rotation=45)
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Bar Plot of Top 10 Word Frequencies')
plt.show()
```