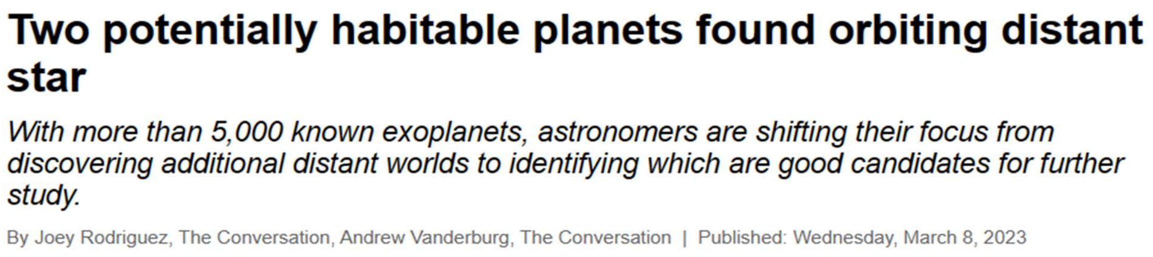
# DOCUMENTATION OF SENTIMENTAL ANALYSIS

The news article that is used for this project is from the link:
“https://astronomy.com/news/2023/03/two-potentially-habitable-planets-found-orbiting-distant-star”.

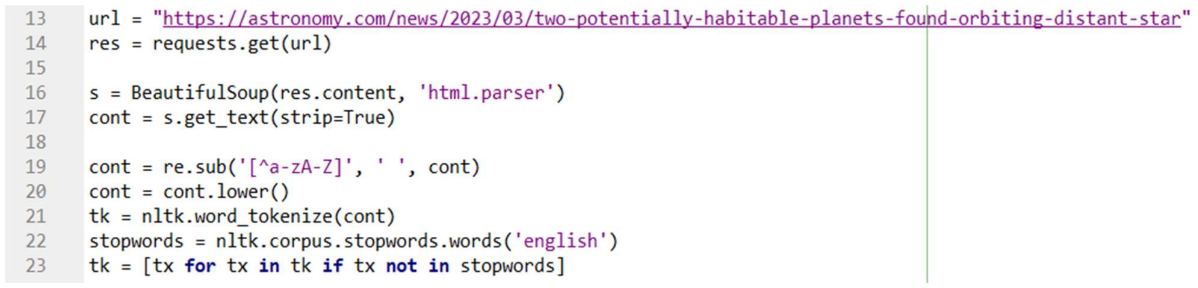


Two potentially habitable planets found orbiting distant star

*With more than 5,000 known exoplanets, astronomers are shifting their focus from discovering additional distant worlds to identifying which are good candidates for further study.*

By Joey Rodriguez, The Conversation, Andrew Vanderburg, The Conversation | Published: Wednesday, March 8, 2023

**THE DATA:**



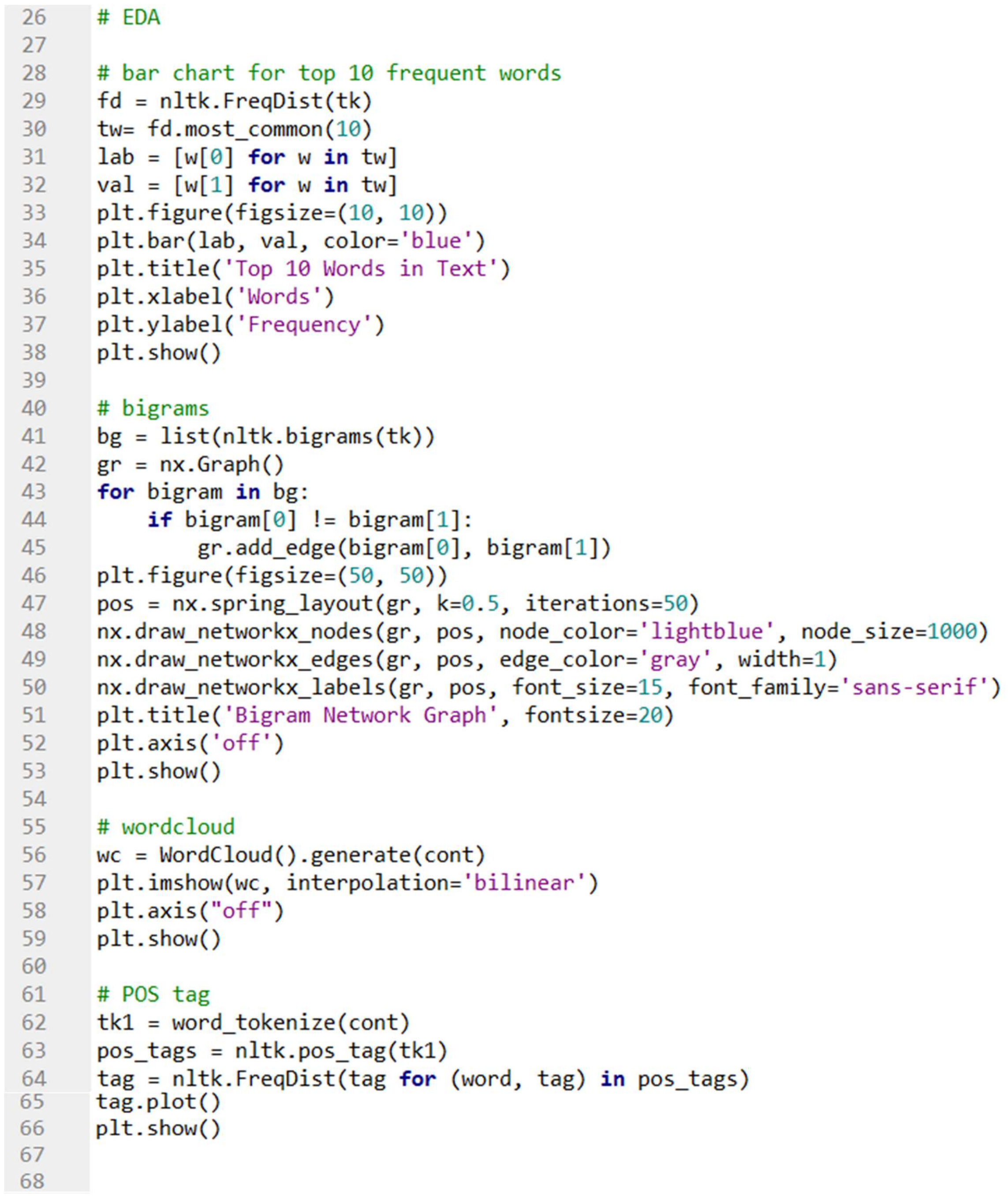
```python
url = "https://astronomy.com/news/2023/03/two-potentially-habitable-planets-found-orbiting-distant-star"
res = requests.get(url)

s = BeautifulSoup(res.content, 'html.parser')
cont = s.get_text(strip=True)

cont = re.sub('[^a-zA-Z]', ' ', cont)
cont = cont.lower()
tk = nltk.word_tokenize(cont)
stopwords = nltk.corpus.stopwords.words('english')
tk = [tx for tx in tk if tx not in stopwords]
```

First the url is set and request is set in res. Then,

- **s = BeautifulSoup(res.content, 'html.parser'**): This line uses the BeautifulSoup library to parse the HTML content of the webpage that was retrieved using the requests library. The parsed content is stored in the variable s.
- **cont = s.get_text(strip=True):** This line extracts the text from the parsed HTML content stored in s using the get_text method of the BeautifulSoup object. The strip=True argument removes any leading or trailing white space from the text.
- **cont = re.sub('[^a-zA-Z]', ' ', cont):** This line uses the re library to substitute any character that is not a letter (i.e., anything that is not in the range a-z or A-Z) with a space. This effectively removes any punctuation, numbers, or other non-letter characters from the text.
- **cont = cont.lower():** This line converts all the remaining letters in cont to lowercase. This is often done to simplify text analysis by reducing the number of unique words that need to be considered.
- **tk = nltk.word_tokenize(cont):** This line uses the word_tokenize function from the nltk library to split the text into a list of individual words (tokens).
- **stopwords = nltk.corpus.stopwords.words('english'):** This line loads a list of common English stopwords from the nltk library. Stopwords are words that are considered to be uninformative or redundant for text analysis purposes, such as "the", "a", and "an".

- **tk = [tx for tx in tk if tx not in stopwords]:** This line creates a new list of tokens (tk) that excludes any tokens that appear in the stopwords list. This helps to further clean the text data by removing uninformative words.

*EDA :*

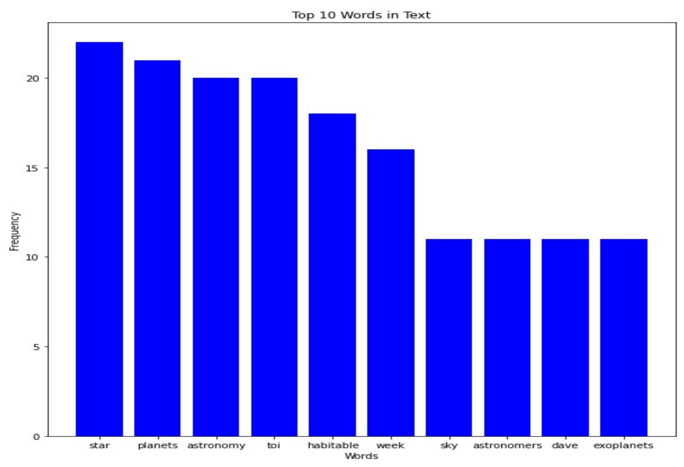


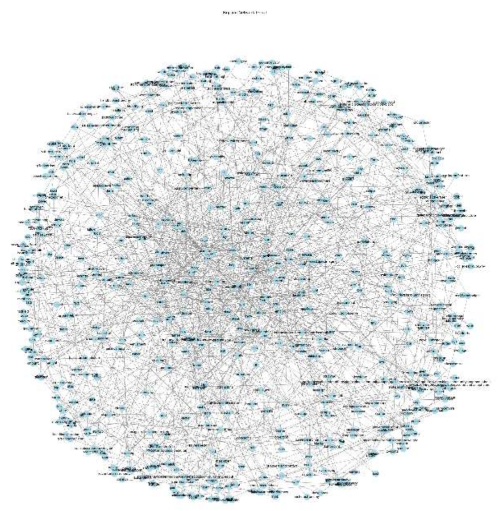
```python
# EDA

# bar chart for top 10 frequent words
fd = nltk.FreqDist(tk)
tw= fd.most_common(10)
lab = [w[0] for w in tw]
val = [w[1] for w in tw]
plt.figure(figsize=(10, 10))
plt.bar(lab, val, color='blue')
plt.title('Top 10 Words in Text')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.show()

# bigrams
bg = list(nltk.bigrams(tk))
gr = nx.Graph()
for bigram in bg:
    if bigram[0] != bigram[1]:
        gr.add_edge(bigram[0], bigram[1])
plt.figure(figsize=(50, 50))
pos = nx.spring_layout(gr, k=0.5, iterations=50)
nx.draw_networkx_nodes(gr, pos, node_color='lightblue', node_size=1000)
nx.draw_networkx_edges(gr, pos, edge_color='gray', width=1)
nx.draw_networkx_labels(gr, pos, font_size=15, font_family='sans-serif')
plt.title('Bigram Network Graph', fontsize=20)
plt.axis('off')
plt.show()

# wordcloud
wc = WordCloud().generate(cont)
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()

# POS tag
tk1 = word_tokenize(cont)
pos_tags = nltk.pos_tag(tk1)
tag = nltk.FreqDist(tag for (word, tag) in pos_tags)
tag.plot()
plt.show()


```

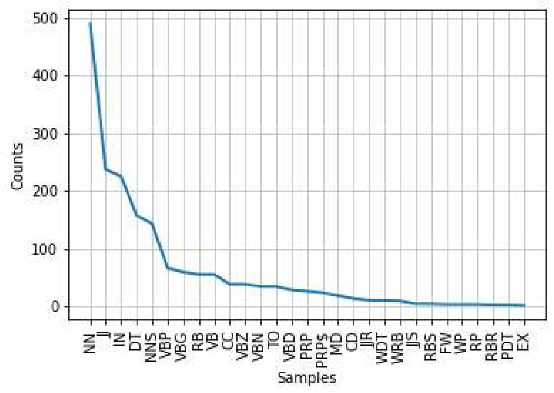
- Bar chart for top 10 frequent words: This graph helps in identifying the most frequent words in the text, which can give an idea about the main topics or themes present in the text.

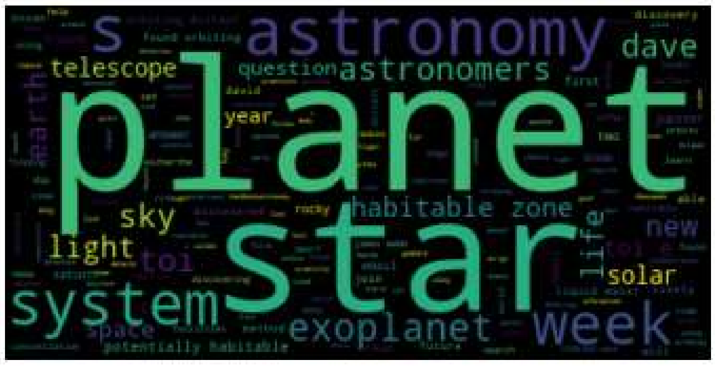
- Bigram network graph: This graph shows the relationships between the bigrams (two-word phrases) in the text, which can give insights into the co-occurrence patterns and semantic relationships between words.
- Wordcloud: This graph provides a visual representation of the most frequent words in the text, with the size of each word indicating its frequency. It can help in identifying the most prominent themes or topics in the text at a glance.
- Part-of-speech (POS) tag plot: This graph shows the distribution of different parts of speech (nouns, verbs, adjectives, etc.) in the text, which can give insights into the syntactic structure and complexity of the text.

**GRAPHS:**

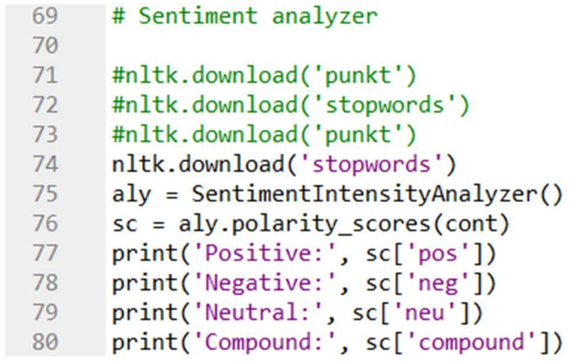








**SENTIMENTAL ANALYSIS :**

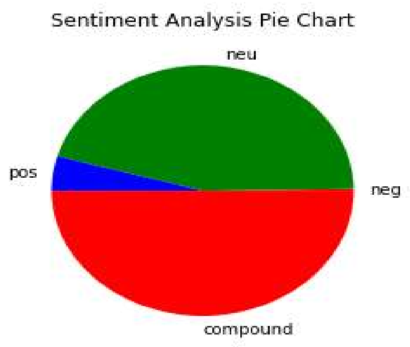


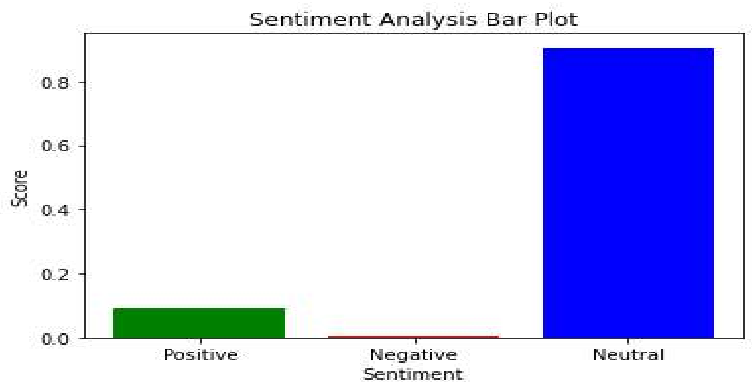
```
# Sentiment analyzer

#nltk.download('punkt')
#nltk.download('stopwords')
#nltk.download('punkt')
nltk.download('stopwords')
aly = SentimentIntensityAnalyzer()
sc = aly.polarity_scores(cont)
print('Positive:', sc['pos'])
print('Negative:', sc['neg'])
print('Neutral:', sc['neu'])
print('Compound:', sc['compound'])
```

```python

# Visualize the sentiment

# Pie chart
plt.pie(sc.values(), labels=sc.keys(), colors=['red', 'green', 'blue'])
plt.title('Sentiment Analysis Pie Chart')
plt.show()


# Bar Plot
sentiment_scores = {
    'Positive': sc['pos'],
    'Negative': sc['neg'],
    'Neutral': sc['neu']
}
labels = list(sentiment_scores.keys())
values = list(sentiment_scores.values())
plt.bar(labels, values, color=['green', 'red', 'blue'])
plt.title('Sentiment Analysis Bar Plot')
plt.xlabel('Sentiment')
plt.ylabel('Score')
plt.show()
```

- The code analyzes the sentiment of a given text using the VADER (Valence Aware Dictionary and sEntiment Reasoner) algorithm.
- The sentiment analyzer is created by importing the SentimentIntensityAnalyzer class from the nltk.sentiment.vader module.
- The analyzer object is used to compute the polarity scores of the given text by calling the polarity_scores() method on it. The method returns a dictionary of scores for positive, negative, neutral, and compound sentiment.
- The scores are then printed to the console to show the relative strength of positive, negative, and neutral sentiment in the text. The compound score is also printed, which is a single value that represents an overall sentiment score ranging from -1 (most negative) to +1 (most positive).
- The code then visualizes the sentiment using two different plots: a pie chart and a bar plot.
  For the pie chart, the sentiment scores are passed to the plt.pie() method to create a pie chart showing the relative strength of each sentiment type.
  For the bar plot, a dictionary of sentiment scores is created, and the scores are plotted using the plt.bar() method to create a bar chart showing the relative strength of each sentiment type.

**GRAPHS:**


Sentiment Analysis Pie Chart
neu
pos
neg
compound


Sentiment Analysis Bar Plot
Score
Positive
Negative
Neutral
Sentiment