Bahrain Polytechnic بوليتكنك البحرين

MSc in AI - Assessment Cover Sheet

Complete and attach this cover sheet to your assessment before submitting

Course Code and Title:	IT9002– Natural Language Processing
Assessment Title:	Individual Project

Learning Outcomes:

LO1 – Understanding the critical knowledge of fundamental concepts, algorithms, and models in Natural Language processing (NLP) for performing various linguistic NLP tasks.

LO2 - Demonstrate professional levels of insight, interpretation by utilizing the various NLP packages like Natural Language Tool Kit (NLTK) to apply, solve, implement, evaluate, and improve the real time significant applications of NLP

Student IDs:	202306805
Student Names:	Maryam Shaaban
Tutor:	Sini Raj Pulari
Individual Project Submission Date	**Tuesday,19th December** 2023 by 11:55 p.m.
Late Rule:	**The maximum grade granted for late submission is 60 % for up to 3 calendar days. A grade of 0 will be allocated for submission after 3 days**

By submitting this assessment for marking, either electronically or as hard copy, I confirm the following:

- *This assignment is **our own work***
- *Any information used has been properly referenced.*
- *I understand that a copy of my work may be used for moderation.*
- *I have kept a copy of this assignment*

Do not write below this line. For Polytechnic use only.

Assessor:	**Date of Marking:**
Grade/Mark:	

Table of Content

Refresh

Task 1: Problem Statment Formulation and Defenition

The term “fake news” refers to false/misleading information that is published by conventional media platforms such as television and unconventional media platforms and social media that includes falsehoods. The target of disseminating false information is to profit from media hype, deceive readers, and affect an organization's reputation. Furthermore, media can create propaganda and can be considered a tool to control people’s thinking, freedom, and perspectives.

Nowadays, in our digital era, fake information will spread viciously leading to multidimensional and substantial threats. The following influences increase the strain of the issue of publicizing false information:

1. Effect on public discourse.
2. Spread of false information.
3. Controversies with Detection.
4. Lack of trust.
5. Social media and algorithms' role.

To address false news issues, we need to develop a comprehensive strategy encompassing media technologies, media literacy, regulatory frameworks, and cooperative international efforts. A collaboration among Governments, technical entities, educators, and the public, the negative effects on our society can be addressed and prevented. Moreover, this initiative focuses on the technological approach to identify and classify fake news and articles thus, ensuring the improvement of media literacy, enabling fact-checking, and protecting the public from misleading information.

1.2 Problem Statement

In this project, natural language processing subtasks are used for “Fake News Detection”, where any text is sorted out by classifying it as real/fake. Hence, the target is to start developing algorithms that can detect fake information and label it. This technology can be used to prevent misleading and fake information.

How can a machine surpass humans in processing and identifying fake news? There are two qualities in machines that aid this process, the first is the ability to keep track of and analyze statistics. The second is the ability to search larger data or knowledge bases for relevant information and respond based on various sources promptly compared to a human. The focus of this project can be illustrated by the following question:

- Can we accurately identify fake news with the aid of machine learning techniques?

Despite the two discussed machine qualities, the aim here is to use supervised learning to train a machine to detect fake data or information, by identifying content and language features specific to the sources without referring to or searching knowledge bases or fact-check tools.

1.3 Expected Resaults

To deal with the detection of fake or real news, the project will be developed in Python with the aid of numerous libraries and natural language processing technologies mainly NLTK and spacy. It is necessary to develop a machine learning model in order to consistently identify the news collection as genuine or false.

First news data, will be gathered from an internet dataset. Then it will be processed using multiple techniqes and will be represented in suitable forms. Next, fit the model, transform, and initialize the classifier. Once the performance matrices have been calculated, the model's performance will be evaluated by computing the relevant performance matrix or matrices.

Required Setup and Libraries for this Project

Libraries Installation

```
# Installation of required libraries
!pip install spacy
!pip install nltk
!pip install wordcloud
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.10/dist-packages (3.6.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in /usr/local/lib/python3.10/dist-packages (from spacy) (8.1.12)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.9.0)
Requirement already satisfied: pathy>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.10.3)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.10/dist-packages (from spacy) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (4.66.1)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.23.5)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.10.13)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.1.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy) (67.7.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (23.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.3.0)
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.0.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2023
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8->spacy) (0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8->spacy) (0.
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.10.0,>=0.3.0->spacy) (8.1.7)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy) (2.1.3)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-packages (1.9.2)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.10/dist-packages (from wordcloud) (1.23.5)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (4.46.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1
```

Libraries Import

```
# Pandas: Data manipulation and analysis library
import pandas as pd

# numpy numerical computing library for arrays, matrices, and mathematical operations
import numpy as np

# Matplotlib: Plotting library for creating visualizations
import matplotlib.pyplot as plt
#import seaborn library for creating confusion matrix visualization
import seaborn as sns

# String: Text processing modules for string operations
import string

# Spacy: Advanced Natural Language Processing (NLP) library
import spacy

# NLTK: Natural Language Toolkit for text processing tasks
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer

# Gensim: Library for topic modeling, document indexing, and similarity retrieval
import gensim
from gensim.models import Word2Vec

#import world cloud
from wordcloud import WordCloud

# Import counter
from collections import Counter

#import defaultdict from collections
from collections import defaultdict

#Model training Libraries
# sklearn necessary modules for text feature extraction and transformation
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
# sklearn.model_selection: Provides train/test split function
from sklearn.model_selection import train_test_split

# Classification libraries
# linearSVC (Linear Support Vector Classifier) from the scikit-learn library
from sklearn.svm import LinearSVC
# LogisticRegression model from the scikit-learn library
from sklearn.linear_model import LogisticRegression

#Evaluation Libraries
# sklearn.metrics for model performance analysis
from sklearn import metrics
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

Mount Google drive if you are using Google Colab

```
# Import the drive module from the google.colab library and mount it.
from google.colab import drive
drive.mount('/content/drive')
```

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

Task 2: Data Collection

To read and save the news data from a CSV file, a data frame is created in this part. After that, some data exploration will be done out in order to gain some insights.The dataset in used for this project is a Kaggle online available dataset. The dataset link:
https://www.kaggle.com/datasets/rajatkumar30/fake-news

2.1 Read Data File

```
# import pandas library for data manipulation and analysis
import pandas as pd
# create the dataframe to store the data from CSV file
df = pd.read_csv(r"/content/drive/MyDrive/Colab Notebooks/Projects/news.csv")
```

This dataset is chosen for this analysis becasue is readily accessible and conveniently formatted for analysis on Kaggle. It was selected due to its direct relevance to the central question: "Can machine learning methods accurately identify fake news?" Furthermore, this dataset includes labeled information, crucial for supervised learning tasks, aligning well with the problem's objectives. Specifically, the dataset comprises labeled articles categorized as either real or fake, accurately representing the target variable.

2.2 Data Analysis

```
# display the 5 top rows from the dataset
df.head()
```

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

```
#show the information of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6335 entries, 0 to 6334
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  6335 non-null   int64 
 1   title       6335 non-null   object
 2   text        6335 non-null   object
 3   label       6335 non-null   object
dtypes: int64(1), object(3)
memory usage: 198.1+ KB
```

The dataset contains the following columns:

- **Unnamed: 0:** An integer column that serves as the articles' index or sequence number with 6335 integer values are non-null in it.
- **title:** Textual data (object type) in a column that represents the entries' titles with 6335 non-null string values.
- **text:** An additional column with textual data (object type) that serves as the entry's main body or text content with 6335 non-null string values .
- **label:** An object column that most likely corresponds to the labels or categories given to each item and contains categorical data (object type) with 6335 non-null string values.

```
#show the shape of the data
df.shape
```

```
(6335, 4)
```

This dataset comprises 6335 rows and 4 columns, as indicated by the data shape (6335, 4). In a dataset, each row usually corresponds to a distinct observation (in this case, a news story). and the characteristics of the article (text, label, index number, and title) are listed in each column.

The dataset does not contain any null values.The index number of the articles is stored in an unnamed column in this dataset; however, it is not crucial to the classification process, so you are free to rename the column or remove it from the data frame.

This is somehow a large dataset so, it will take alot of memory space and time to preform the text preprocessing, representation and classification on the body of each news article. However, for this project the text column will be merged with title column to create the article column as the text source.

```
# Moderation that is necessary for the classification process
# Deleting the unnameed column which holds an integer values that are reffering to the index
df = df.drop(['Unnamed: 0'], axis=1)
# Creating an article column which combines the title and the text of the news
df['article'] = df['title'] + df['text']
# Show the first 5 rows of the data frame
df.head()
```

	title	text	label	article
0	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE	You Can Smell Hillary's FearDaniel Greenfield,...
1	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE	Watch The Exact Moment Paul Ryan Committed Pol...
2	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL	Kerry to go to Paris in gesture of sympathyU.S...
3	Bernie supporters on Twitter erupt in anger	— Kaydee King (@KaydeeKing) November 9,	FAKE	Bernie supporters on Twitter erupt in anger

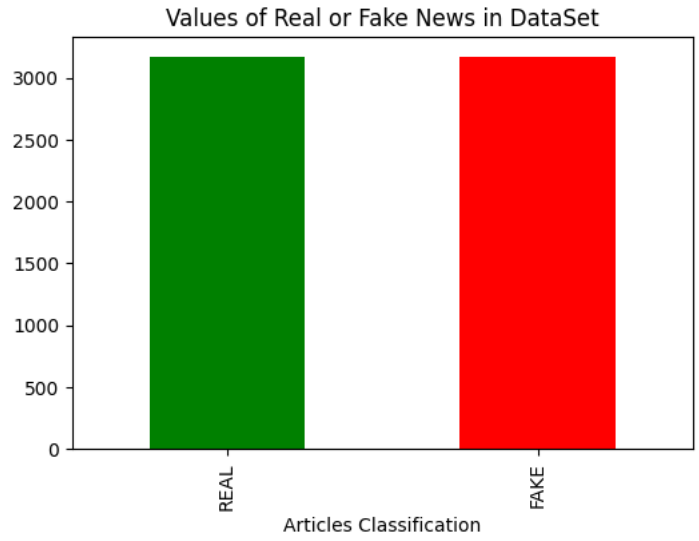
```
# Checking how the data is distributed per label of news authenticity
df["label"].value_counts()
```

```
REAL    3171
FAKE    3164
Name: label, dtype: int64
```

In terms of the classes that the labels "REAL" and "FAKE" represent, this distribution indicates that the dataset is reasonably balanced. There are comparatively equal amounts of occurrences in both classes with 3164 instances of "FAKE" and 3171 instances of "REAL".

Many machine learning techniques benefit from balanced datasets because they minimize biases toward a particular class during model training. That being said, in many situations the model's performance might not be greatly affected by the small discrepancy of 7 instances between "REAL" and "FAKE".

```
# Plot the classes using matplotlib pyplot
plt.figure(figsize=(6, 4))
count_Class=pd.value_counts(df["label"], sort= True)
count_Class.plot(kind = 'bar',color = ["green","red"])
# Set the title and labels of the plot
plt.title('Values of Real or Fake News in DataSet')
plt.xlabel('Articles Classification')
# Display the plot
plt.show()
```



We can see here that the data are almost equally distributed between fake and real news so the data is somehow balanced and should not have any imbalanced observations. However this categorical label column should be changed into numbers.

```
#Change the categorical labels into numbers in the dataset
df['label'] = df.label.map({'REAL':1, 'FAKE':0}) # Real is 1, Fake is 0
# Show the first rows of the dataset
df.head()
```

	title	text	label	article
0	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	0	You Can Smell Hillary's FearDaniel Greenfield,...
1	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	0	Watch The Exact Moment Paul Ryan Committed Pol...
2	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	1	Kerry to go to Paris in gesture of sympathyU.S...
3	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	0	Bernie supporters on Twitter erupt in anger ag...

Task3: Data Preprocessing

Import necessary Libraries to clean text

```
# import Natural Language Toolkit for text processing tasks
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer

# import TF-IDF Vectorizer from sklearn.feature_extraction.text for text feature extraction
from sklearn.feature_extraction.text import TfidfVectorizer

# import string for text processing and string operations
import string
```

Download Required processin packages from NLTK

```
# download the NLTK tokenizer resource 'punkt'
nltk.download('punkt')  # necessary for word tokenization

# download the NLTK stopwords corpus
nltk.download('stopwords')  # set of stopwords for text filtering

# download the NLTK WordNet resource
nltk.download('wordnet')  # for English words lemmatization
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

3.1 Tokenization and cleaning

This function will preprocess the text where it will devide each title into tokens, convert the tokens into lowercase, remove stopword, remove punctuation and special characters

```
# Function for text preprocessing
def preprocess_text(text_data):
    processed_texts = []
    stopwords_set = set(stopwords.words('english'))

    for article in text_data:
        # Tokenization
        tokens = word_tokenize(article)

        # Lowercase conversion
        tokens_lower = [token.lower() for token in tokens]

        # Stopwords removal
        tokens_no_stopwords = [token for token in tokens_lower if token not in stopwords_set]

        # Punctuation removal
        tokens_no_punctuation = [token for token in tokens_no_stopwords if token not in string.punctuation]

        # Remove punctuation and special characters
        tokens_no_special_chars = [token for token in tokens_no_punctuation if token.isalnum()]  # Keeps only alphanumeric tokens

        # Combine tokens into a processed text
        processed_text = ' '.join(tokens_no_special_chars)
        processed_texts.append(processed_text)

    return processed_texts

# Perform text preprocessing on the article text and place the cleaned text in the same c
processed_text_data = preprocess_text(df['article'])
df['article'] = processed_text_data
df.head()
```

	title	text	label	article
0	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	0	smell hillary feardaniel greenfield shillman j...
1	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	0	watch exact moment paul ryan committed politic...
2	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	1	kerry go paris gesture secretary state john ke...
3	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	0	bernie supporters twitter erupt anger dnc trie...

The results of cleaning the text are shown in the article column, where you can clearly see the difference between the text in it and the text in the (title and text) columns.

3.2 Lemmatization

```
# Initialize WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

# Function for lemmatizing words
def lemmatize_words(text):
    return " ".join([lemmatizer.lemmatize(word) for word in text.split()])

# Apply lemmatization to the 'article' column
df["article"] = df["article"].apply(lambda text: lemmatize_words(text))
df.head()
```

	title	text	label	article
0	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	0	smell hillary feardaniel greenfield shillman j...
1	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	0	watch exact moment paul ryan committed politic...
2	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	1	kerry go paris gesture secretary state john ke...
3	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	0	bernie supporter twitter erupt anger dnc tried...

While both lemmatization and stemming are methods used to reduce words to their base or root form, their methods are not the same. in this project the lemmatization was utilized because of the need to preserve context of the articles. Also, Lemmatization, as opposed to stemming,

often produces more accurate words since it takes into account the morphological analysis of words and maps them to a dictionary.

3.3 TF-IDF Vectorization

```
# Import TF-IDF Vectorizer for text feature extraction
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TF-IDF Vectorization
vectorizer = TfidfVectorizer()

# Convert the text into a numrical matrix using the vectorizer
x = vectorizer.fit_transform(df['article'])

# Get feature names
feature_names = vectorizer.get_feature_names_out()

# Display TF-IDF matrix and feature names
print("TF-IDF matrix:")
print(x.toarray())
print("\nFeature names:")
print(feature_names)
```

```
TF-IDF matrix:
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]

Feature names:
['00' '000' '007' ... 'کدآمایی' 'والمرضی' 'هذاposted']
```

A crucial stage in NLP preprocessing is TF-IDF vectorization, which converts unstructured text input into a useful and structured format that is appropriate for machine learning models. This allows for better analysis and helps the models comprehend and learn from textual data. Since the dataset is considered to be large, the TF-IDF Matrix represents these results. However, by reducing the effect of articles length differences, it makes comparing documearticlesnts of various lengths possible. In order to guarantee proper analysis and fair comparisons among articles, this normalization is necessary.

Task 4: Text Representation

Import necessary Libraries to represent text

```
# Import NLTK processing tasks
import nltk
from nltk.tokenize import word_tokenize
from nltk.util import ngrams # necessary for N grams

# Import spacy processing library for Named Entity Recognision
import spacy

# Import gensim library
from gensim.models import Word2Vec

# Import wordcloud
from wordcloud import WordCloud

# Import counter
from collections import Counter
```

Download representation packages from nltk and spacy

```
spacy.cli.download("en_core_web_sm") # necessary for NER
nltk.download('averaged_perceptron_tagger') # necessary for POS
nltk.download('punkt') # necessary for word tokenizatio
```

```
✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

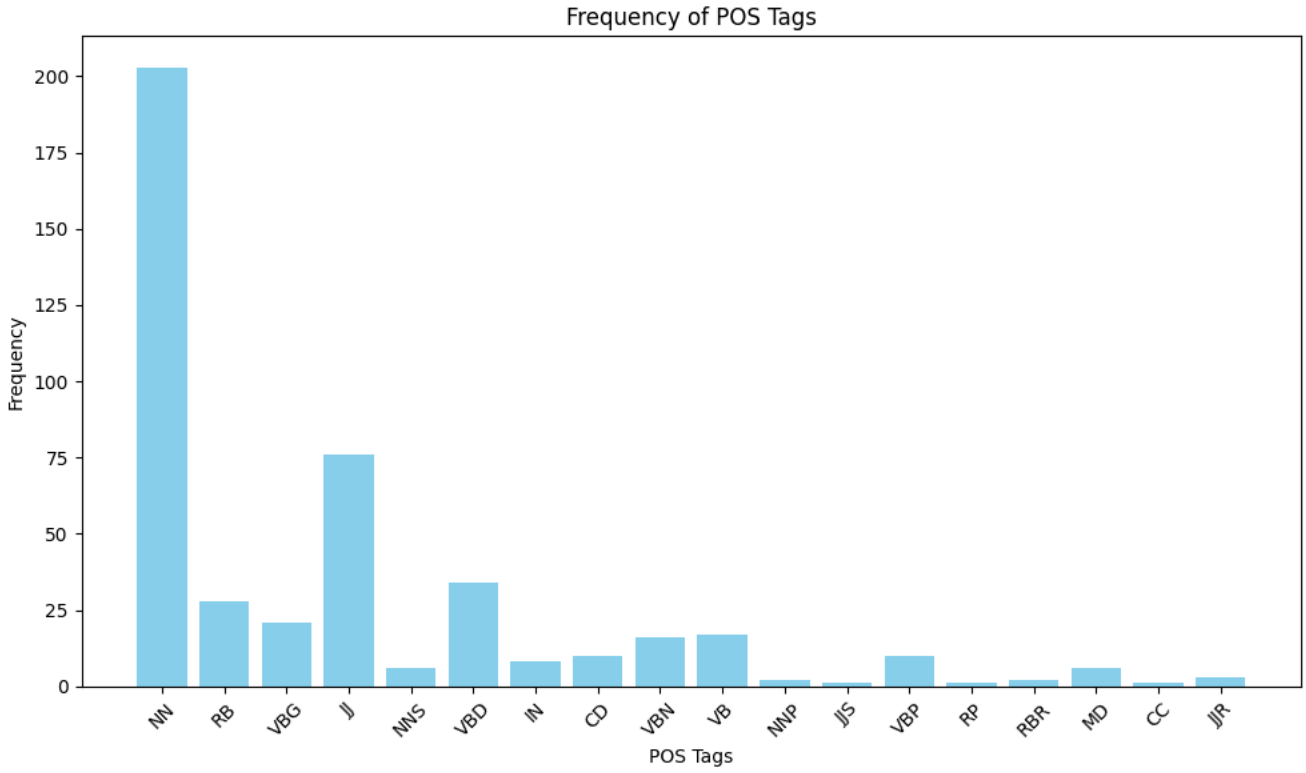
4.1 Part of Speech Tagging

```
# Create a function for POS tagging using NLTK
def pos_tagging(text_data):
    # Initializar an empty list to store part of speech tagging
    pos_tags = []
    # Iterate through each article in the dataset
    for news_article in text_data:
      # Tokenize the article text
        tokens = word_tokenize(news_article)
      # Pos tag the tokens
        tagged_words = nltk.pos_tag(tokens)
        #Count the POS tags
        the_count = Counter(tag for _, tag in tagged_words)
        # Add the tagged tokens to the list
        pos_tags.append(tagged_words)
    # Return the list
    return pos_tags, the_count

# Perform POS tagging using NLTK
pos_tags_result , the_count = pos_tagging(df['article'])

print(the_count)

# Print the POS results of the articles in the dataset
print("POS tags using NLTK:")
for idx, tags in enumerate(pos_tags_result):
    print(f"POS tags for article {idx + 1}:")
    print(tags)
```

```
Counter({'NN': 161, 'JJ': 73, 'NNS': 41, 'VBD': 31, 'RB': 28, 'VBG': 21, 'VBP': 20, 'VBN': 16, 'VB': 15, 'CD': 9, 'IN': 8, 'VBZ': 7,
POS tags using NLTK:
POS tags for article 1:
[('smell', 'NN'), ('hillary', 'JJ'), ('feardaniel', 'NN'), ('greenfield', 'VBD'), ('shillman', 'JJ'), ('journalism', 'NN'), ('fellow
POS tags for article 2:
[('watch', 'NN'), ('exact', 'JJ'), ('moment', 'NN'), ('paul', 'NN'), ('ryan', 'NN'), ('committed', 'VBD'), ('political', 'JJ'), ('sui
POS tags for article 3:
[('kerry', 'NN'), ('go', 'VBP'), ('paris', 'JJ'), ('gesture', 'NN'), ('secretary', 'NN'), ('state', 'NN'), ('john', 'NN'), ('kerry',
POS tags for article 4:
[('bernie', 'NN'), ('supporters', 'NNS'), ('twitter', 'VBP'), ('erupt', 'JJ'), ('anger', 'NN'), ('dnc', 'NN'), ('tried', 'VBD'), ('wa
POS tags for article 5:
[('battle', 'NN'), ('new', 'JJ'), ('york', 'NN'), ('primary', 'JJ'), ('mattersit', 'NNS'), ('primary', 'JJ'), ('day', 'NN'), ('new',
POS tags for article 6:
[('tehran', 'NN'), ('usa', 'JJ'), ('immigrant', 'JJ'), ('grandparents', 'NNS'), ('50', 'CD'), ('years', 'NNS'), ('ago', 'RB'), ('arri
POS tags for article 7:
[('girl', 'NN'), ('horrified', 'VBD'), ('watches', 'NNS'), ('boyfriend', 'VBP'), ('left', 'VBD'), ('facetime', 'JJ'), ('onshare', 'JJ
POS tags for article 8:
[('britain', 'NN'), ('schindler', 'NN'), ('dies', 'VBZ'), ('106a', 'CD'), ('czech', 'NN'), ('stockbroker', 'NN'), ('saved', 'VBD'),
POS tags for article 9:
[('fact', 'NN'), ('check', 'VB'), ('trump', 'IN'), ('clinton', 'NN'), ('forumhillary', 'JJ'), ('clinton', 'NN'), ('donald', 'NN'), (
POS tags for article 10:
[('iran', 'NN'), ('reportedly', 'RB'), ('makes', 'VBZ'), ('new', 'JJ'), ('push', 'JJ'), ('uranium', 'NN'), ('concessions', 'NNS'), (
POS tags for article 11:
[('three', 'CD'), ('clintons', 'NNS'), ('iowa', 'VBP'), ('glimpse', 'NN'), ('fire', 'NN'), ('eluded', 'VBD'), ('hillary', 'JJ'), ('cl
POS tags for article 12:
[('donald', 'JJ'), ('trump', 'NN'), ('shockingly', 'RB'), ('weak', 'JJ'), ('delegate', 'JJ'), ('game', 'NN'), ('somehow', 'NN'), ('go
POS tags for article 13:
[('strong', 'JJ'), ('solar', 'JJ'), ('storm', 'NN'), ('tech', 'NN'), ('risks', 'NNS'), ('today', 'NN'), ('s0', 'VBP'), ('news', 'NN')
POS tags for article 14:
[('10', 'CD'), ('ways', 'NNS'), ('america', 'RB'), ('preparing', 'VBG'), ('world', 'NN'), ('war', 'NN'), ('3october', 'CD'), ('31',
POS tags for article 15:
```

```
[('trump', 'NN'), ('takes', 'VBZ'), ('cruz', 'JJ'), ('lightlykilling', 'VBG'), ('obama', 'JJ'), ('administration', 'NN'), ('rules',
POS tags for article 16:
[('women', 'NNS'), ('lead', 'VBP'), ('differentlyas', 'JJ'), ('women', 'NNS'), ('move', 'VBP'), ('high', 'JJ'), ('offices', 'NNS'),
POS tags for article 17:
[('shocking', 'VBG'), ('michele', 'JJ'), ('obama', 'JJ'), ('hillary', 'JJ'), ('caught', 'NN'), ('glamorizing', 'VBG'), ('date', 'NN')
POS tags for article 18:
[('hillary', 'JJ'), ('clinton', 'NN'), ('huge', 'JJ'), ('trouble', 'NN'), ('america', 'NN'), ('noticed', 'VBD'), ('sick', 'JJ'), ('th
POS tags for article 19:
[('iran', 'NN'), ('bill', 'NN'), ('obama', 'IN'), ('like', 'IN'), ('washington', 'NN'), ('cnn', 'NN'), ('months', 'NNS'), ('white',
POS tags for article 20:
[('1', 'CD'), ('chart', 'NN'), ('explains', 'VBZ'), ('everything', 'NN'), ('need', 'NN'), ('know', 'VBP'), ('partisanship', 'NN'), (
POS tags for article 21:
[('slippery', 'NN'), ('slope', 'NN'), ('trump', 'NN'), ('proposed', 'VBN'), ('ban', 'NN'), ('muslimswith', 'NN'), ('little', 'JJ'),
POS tags for article 22:
[('episode', 'NN'), ('160', 'CD'), ('sunday', 'NN'), ('wire', 'NN'), ('hail', 'NN'), ('deplorables', 'VBZ'), ('special', 'JJ'), ('gue
POS tags for article 23:
[('hillary', 'JJ'), ('clinton', 'NN'), ('makes', 'VBZ'), ('bipartisan', 'JJ'), ('appeal', 'JJ'), ('staten', 'VBN'), ('islandhillary',
POS tags for article 24:
[('new', 'JJ'), ('senate', 'JJ'), ('majority', 'NN'), ('leader', 'NN'), ('main', 'JJ'), ('goal', 'NN'), ('gop', 'NN'), ('scarymitch',
POS tags for article 25:
[('inferno', 'JJ'), ('overpopulation', 'NN'), ('november', 'RB'), ('1', 'CD'), ('2016', 'CD'), ('inferno', 'NN'), ('great', 'JJ'), (
POS tags for article 26:
[('forces', 'NNS'), ('seek', 'VBP'), ('delegate', 'NN'), ('revoltwashington', 'NN'), ('cnn', 'NN'), ('faction', 'NN'), ('gop', 'NN'),
POS tags for article 27:
[('sanders', 'NNS'), ('trounces', 'NNS'), ('clinton', 'VBP'), ('make', 'VB'), ('difference', 'NN'), ('meanwhile', 'RB'), ('democrat',
POS tags for article 28:
```

```
# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(labels, values, color='skyblue')
plt.xlabel('POS Tags')
plt.ylabel('Frequency')
plt.title('Frequency of POS Tags')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability if needed
plt.tight_layout()
plt.show()
```



A key step in representing the news text in this project is Part of Speech tagging, which is assigning words in the phrase to the appropriate parts of speech, such as nouns, verbs, adjectives, adverbs.. etc . POS tagging is crucial for a number of NLP and language analysis applications, however in this project it important in terms of recognizing sentence structure, and examining articles' language styles, word choices, and grammatical structures, which might provide hints about the reliability of the source. The use of speech fragments in an unusual or inconsistent way may cause questions about the veracity of the news source.

Based on the frequency of distinct POS tags, the plot illustrates how different parts of speech are distributed within the examined articles in the dataset, offering insights into the linguistic composition and qualities of the text.

4.2 Named Entity Recognition

```
# Create a function for NER using spacy
def named_entity_rec(text_data):
    # Load the small english model from the library
    nlp = spacy.load("en_core_web_sm")
    # Initialize an empty list to store named entity recognition
    ner_tags = []
    # Initialize a counter to count ner
    entity_counter = Counter()
    # Iterate through each article in the dataset
    for news_article in text_data:
      # Process the text from the article and extract named entity with the corresponding text
        art = nlp(news_article)
        #Count NER
        ner_tags.append([(ent.text, ent.label_) for ent in art.ents])
    # Return the text list containing the NER
    return ner_tags

# Perform NER using spacy
ner_tags_result = named_entity_rec(df['article'])

# Print the NER results of the articles in the dataset
print("Named Entity Recognition using spaCy:")
for idx, tags in enumerate(ner_tags_result):
    print(f"NER tags for atricle {idx + 1}:")
    print(tags)
```

```
Named Entity Recognition using spaCy:
NER tags for atricle 1:
[('hillary feardaniel', 'PERSON'), ('new york', 'GPE'), ('hillary rodham clinton', 'PERSON'), ('fbi', 'ORG'), ('fbi', 'ORG'), ('hilla
NER tags for atricle 2:
[('paul', 'PERSON'), ('google', 'ORG'), ('two', 'CARDINAL'), ('paul', 'PERSON'), ('today', 'DATE'), ('weeks ago', 'DATE'), ('united s
NER tags for atricle 3:
[('kerry', 'PERSON'), ('paris', 'GPE'), ('john kerry', 'PERSON'), ('monday', 'DATE'), ('paris', 'GPE'), ('later week', 'DATE'), ('ame
NER tags for atricle 4:
[('bernie', 'PERSON'), ('november 9 2016', 'DATE'), ('tonight', 'TIME'), ('dem', 'NORP'), ('democrats', 'NORP'), ('bernie', 'PERSON')
NER tags for atricle 5:
[('new york', 'GPE'), ('mattersit primary', 'PERSON'), ('new york', 'GPE'), ('hillary clinton', 'PERSON'), ('republican', 'NORP'), (
NER tags for atricle 6:
[('tehran usa', 'ORG'), ('50 years ago', 'DATE'), ('new york city', 'GPE'), ('iran', 'GPE'), ('playing little league raritan', 'ORG')
NER tags for atricle 7:
[('baylee luciani', 'PERSON'), ('screenshot baylee', 'PERSON'), ('baylee luciani', 'PERSON'), ('austin', 'PERSON'), ('evening', 'TIME
NER tags for atricle 8:
[('britain schindler', 'ORG'), ('106a', 'CARDINAL'), ('czech', 'NORP'), ('650', 'CARDINAL'), ('jewish', 'NORP'), ('nazi germany', 'GP
NER tags for atricle 9:
[('clinton', 'PERSON'), ('clinton', 'PERSON'), ('nbc', 'ORG'), ('clinton', 'PERSON'), ('iraq', 'GPE'), ('iraq', 'GPE'), ('obama', 'GP
NER tags for atricle 10:
[('iran', 'GPE'), ('five', 'CARDINAL'), ('iran', 'GPE'), ('days', 'DATE'), ('new york times', 'ORG'), ('late sunday', 'DATE'), ('russ
NER tags for atricle 11:
[('three', 'CARDINAL'), ('hillary clinton', 'PERSON'), ('one', 'CARDINAL'), ('1992', 'DATE'), ('bill clinton', 'PERSON'), ('saturday
NER tags for atricle 12:
[('donald trump', 'PERSON'), ('weekend', 'DATE'), ('colorado', 'GPE'), ('gop', 'ORG'), ('republicans', 'NORP'), ('colorado', 'GPE'),
NER tags for atricle 13:
[('today', 'DATE'), ('2012', 'CARDINAL'), ('2016', 'DATE')]
NER tags for atricle 14:
[('10', 'CARDINAL'), ('america', 'GPE'), ('3october 31 2016', 'DATE'), ('american', 'NORP'), ('vietnam', 'EVENT'), ('4', 'CARDINAL')
NER tags for atricle 15:
[('trump', 'PERSON'), ('obama administration', 'ORG')]
NER tags for atricle 16:
[('democratic', 'NORP'), ('senate', 'ORG'), ('night democratic national convention philadelphia', 'ORG'), ('20', 'CARDINAL'), ('senat
NER tags for atricle 17:
[('first', 'ORDINAL'), ('october 27', 'DATE'), ('alex jones', 'PERSON'), ('hillary clinton', 'PERSON'), ('decade ago', 'DATE'), ('whi
NER tags for atricle 18:
[('hillary clinton', 'PERSON'), ('america', 'GPE'), ('news0', 'PERSON'), ('hillary clinton', 'PERSON'), ('hillary clinton', 'PERSON')
NER tags for atricle 19:
[('iran', 'GPE'), ('bill obama', 'PERSON'), ('washington', 'GPE'), ('cnn', 'ORG'), ('months', 'DATE'), ('white house', 'ORG'), ('cong
NER tags for atricle 20:
[('1', 'CARDINAL'), ('2014', 'DATE'), ('one', 'CARDINAL'), ('american', 'NORP'), ('last 20 years recently 1994', 'DATE'), ('10', 'CAR
NER tags for atricle 21:
[('new york', 'GPE'), ('islamic', 'NORP'), ('2010', 'DATE'), ('zero', 'CARDINAL'), ('pamela geller', 'PERSON'), ('american', 'NORP'),
NER tags for atricle 22:
[('160', 'CARDINAL'), ('sunday', 'DATE'), ('13', 'CARDINAL'), ('160', 'CARDINAL'), ('sunday', 'DATE'), ('november 13 2016', 'DATE'),
NER tags for atricle 23:
```

```
[('hillary clinton', 'PERSON'), ('clinton', 'PERSON'), ('today', 'DATE'), ('senate', 'ORG'), ('republican', 'NORP'), ('george bush',
NER tags for atricle 24:
[('senate', 'ORG'), ('gop', 'ORG'), ('republican', 'NORP'), ('senate', 'ORG'), ('week', 'DATE'), ('senate', 'ORG'), ('two years', 'DA
NER tags for atricle 25:
[('november 1 2016', 'DATE'), ('tom', 'PERSON'), ('robert langdon', 'PERSON'), ('dan brown books', 'PERSON'), ('langdon', 'PERSON'),
NER tags for atricle 26:
[('revoltwashington', 'GPE'), ('cnn', 'ORG'), ('gop', 'ORG'), ('donald trump', 'PERSON'), ('one', 'CARDINAL'), ('two', 'CARDINAL'),
NER tags for atricle 27:
[('clinton', 'PERSON'), ('democrat', 'NORP'), ('two', 'CARDINAL'), ('hillary clinton', 'PERSON'), ('vermont', 'GPE'), ('west virginia
NER tags for atricle 28:
```

Identifying individuals or organizations that are mentioned in news items can help confirm the reliability of the sources. The news may gain more trust if the article cites respectable institutions or authorities. On the other hand, allusions to unidentified or nonexistent entities result in questioning the accuracy of the information.

4.3 BiGrams (Ngrams)

```
# Create a function for Bigram generation
def generate_N_grams(text_data):
    # Initialize an empty list to store generated bigrams
    generated_bigrams = []

    # Iterate through each article in the dataset
    for news_article in text_data:

        # Tokenize the article text
        tokens = word_tokenize(news_article)

        # Generate bigrams using NLTK ngrams funtion
        bigrams = list(ngrams(tokens, 2))

        # Append the bigram to list
        generated_bigrams.append(bigrams)

    # Return the Bigram list
    return generated_bigrams

# Perform bigram using NLTK
bigram_result = generate_N_grams(df['article'])

# Print the bigram results of the articles in the dataset
print("Bigrams:")
for idx, bg in enumerate(bigram_result):
    print(f"Bi-grams for article {idx + 1}:")
    print(bg)


bi_freq_dist = nltk.FreqDist(bigram_result)
```

```
[('war', 'common'), ('common', 'jimmie'), ('jimmie', 'mogila'), ('mogila', 'editor'), ('editor', 'note'), ('note', 'mechanism'), ('m
Bi-grams for article 2131:
[('11', 'benghazi'), ('benghazi', 'takeaway'), ('takeaway', 'one'), ('one', 'hourwashington'), ('hourwashington', 'cnn'), ('cnn', 'ho
Bi-grams for article 2132:
[('julian', 'assange'), ('assange', 'predicts'), ('predicts', 'trump'), ('trump', 'lose'), ('lose', 'still'), ('still', 'missing'),
Bi-grams for article 2133:
[('taxpayer', 'shell'), ('shell', '100k'), ('100k', 'pay'), ('pay', 'cop'), ('cop', 'caught'), ('caught', 'eating'), ('eating', 'weed
Bi-grams for article 2134:
[('bush', 'donor'), ('donor', 'await'), ('await', 'green'), ('green', 'light'), ('light', 'jump'), ('jump', 'shipkilling'), ('shipkil
Bi-grams for article 2135:
[('amnesty', 'intl'), ('intl', 'western'), ('western', 'backed'), ('backed', 'syrian'), ('syrian', 'rebel'), ('rebel', 'must'), ('mus
Bi-grams for article 2136:
[('obama', 'say'), ('say', 'learned'), ('learned', 'clinton'), ('clinton', 'using'), ('using', 'private'), ('private', 'email'), ('em
Bi-grams for article 2137:
[('mccain', 'trump'), ('trump', 'ignore'), ('ignore', 'gop'), ('gop', 'voter'), ('voter', 'cnn'), ('cnn', 'john'), ('john', 'mccain')
Bi-grams for article 2138:
[('trump', 'effect'), ('effect', 'jack'), ('jack', 'box'), ('box', 'obama'), ('obama', 'changethe'), ('changethe', 'trump'), ('trump
Bi-grams for article 2139:
[('disabled', 'veteran'), ('veteran', 'find'), ('find', 'freedom'), ('freedom', 'arnaldo'), ('arnaldo', 'rodgers'), ('rodgers', 'trai
Bi-grams for article 2140:
[('democrat', 'begin'), ('begin', 'search'), ('search', 'candidate'), ('candidate', 'know'), ('know', 'use'), ('use', 'emailwednesday
Bi-grams for article 2141:
[('lesson', 'obama'), ('obama', 'deal'), ('deal', 'iranin'), ('iranin', 'april'), ('april', 'president'), ('president', 'obama'), ('o
Bi-grams for article 2142:
[('hillary', 'failing'), ('failing', 'twitter'), ('twitter', 'today'), ('today', 'happy'), ('happy', 'birthday'), ('birthday', 'futur
Bi-grams for article 2143:
[('early', 'voting'), ('voting', 'could'), ('could', 'favor'), ('favor', 'democrat'), ('democrat', 'key'), ('key', 'stateswith'), ('s
Bi-grams for article 2144:
[('breaking', 'huma'), ('huma', 'abedin'), ('abedin', 'thrown'), ('thrown', 'hillary'), ('hillary', 'campaign'), ('campaign', 'plane
Bi-grams for article 2145:
[('britain', 'longer'), ('longer', 'sovereign'), ('sovereign', 'democracyhome'), ('democracyhome', 'world'), ('world', 'britain'), (
```

Bigrams analyze word pairs that follow one another to offer contextual information. Contextual awareness aids in comprehending the meaning and context of words and phrases used in the text, which is useful when identifying nuances or misleading language used in false news texts.Alsom certain words may have different meanings depending on the words around them in specific settings. Bigrams provide more context, which aids in the disambiguation of such circumstances.

4.4 Word Embedding

```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

# Initialize CountVectorizer
vectorizer = CountVectorizer()

# Fit and transform the text data to bag of words representation
bow_matrix = vectorizer.fit_transform(df['title'])

# Get the feature names (unique words/tokens)
feature_names = vectorizer.get_feature_names_out()

# Convert bag of words matrix to a DataFrame for visualization or further processing
bow_df = pd.DataFrame(bow_matrix.toarray(), columns=feature_names)

# Concatenate the BoW DataFrame with the original DataFrame
result_df = pd.concat([df, bow_df], axis=1)

# Display the resulting DataFrame with Bag of Words representation
result_df.head()
```

	title	text	label	article	00	000	00pm	01	04	05	...	آمریکا	ارزی	القادمون	ایران	جنگ	سال	علیه	مطالعاتی	مورد	کدآمایی
	You Can	Daniel Greenfield, a		smell hillary																	

These embeddings allow machine learning models to comprehend linguistic subtleties by capturing semantic links and contextual information among words by displaying words in a high-dimensional space as dense, significant vectors. They would also help identify word parallels, contrasts, and relationships in false news detection, which helps identify linguistic patterns employed in misleading content. They improve the ability of models to interpret linguistic nuances and capture semantic meanings, which raises the efficacy and accuracy of fake news detection.

1	[illegible] Paul Ryan	[illegible] Reddit	0	paul ryan	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

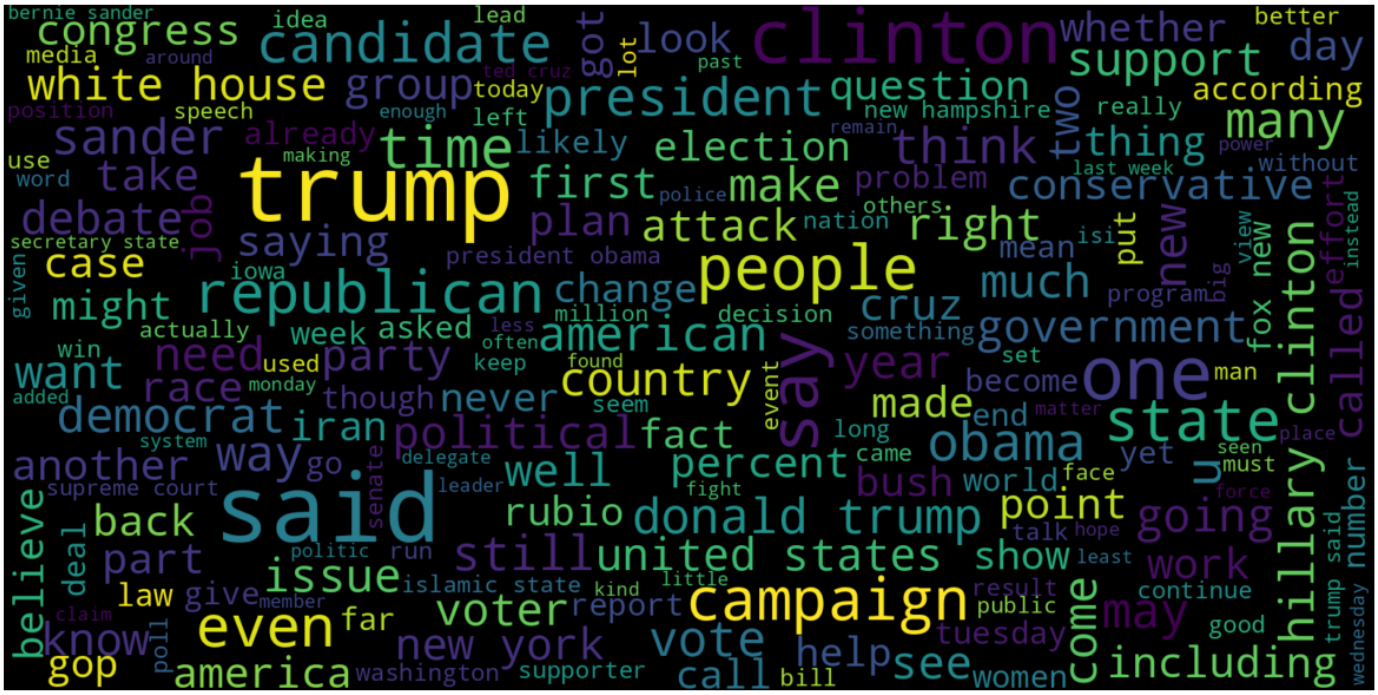
4.5 Word Cloud

kerry go

4.5.1 Real News Cloud

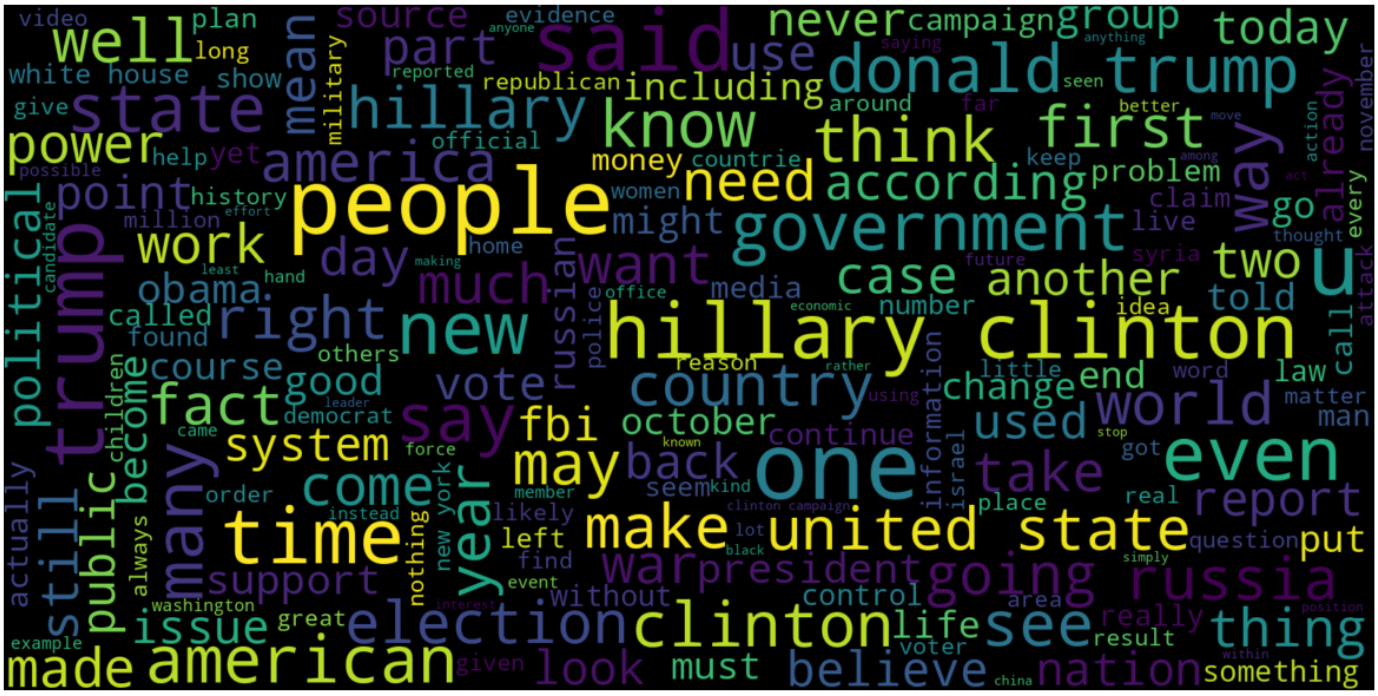
sympathy [illegible] state john

```
# concatenate the text of the real articles
consolidated = ' '.join(word for word in df['article'][df['label'] == 1].astype(str))
# create word cloud object
wordCloud = WordCloud(width=1600, height=800,random_state=21, max_font_size=110)
# configure the figure size
plt.figure(figsize=(15, 10))
# generate a word cloud
plt.imshow(wordCloud.generate(consolidated), interpolation='bilinear')
# hide the access
plt.axis('off')
# print the plot
plt.show()
```



4.5.2 Fake News Cloud

```
# concatenate the text of the fake articles
consolidated = ' '.join(word for word in df['article'][df['label'] == 0].astype(str))
# create word cloud object
wordCloud = WordCloud(width=1600, height=800,random_state=21, max_font_size=110)
# configure the figure size
plt.figure(figsize=(15, 10))
# generate a word cloud
plt.imshow(wordCloud.generate(consolidated), interpolation='bilinear')
# hide the access
plt.axis('off')
# print the plot
plt.show()
```



Task 5 and 6: News Text Classification (Model Training, Testing and Evaluation)

For this project the SKLearn packages were utalized for the "Support Vector Classifier" (SVC) and "Logistic Regression" (LG) models. These models were chosen for the purpose of detecting fake news due to their ability to perform well in tasks involving binary classification.SVC is a flexible tool that works well with the diverse distribution of language data that is frequently used in false news identification. Plus, LG is easily interpretable, making it easier to pinpoint the key characteristics that influence the categorization choice.

Also, both models have record of success in text classification, it makes sense to investigate and set a baseline for the effectiveness of false news detection algorithms using them as trustworthy benchmarks. Their choice for this project is further supported by how simple it is to implement.(reference 1)

Required Libraries for Classification and Evaluation

```
# sklearn necessary modules for text feature extraction and transformation
from sklearn.feature_extraction.text import TfidfTransformer, CountVectorizer, TfidfVectorizer

# Import SVC classifier from sklearn
from sklearn.svm import LinearSVC
# Import LogisticRegression classifier from sklearn
from sklearn.linear_model import LogisticRegression

# Import Confusion matrix from sklearn
from sklearn.metrics import confusion_matrix
```

5.1 Model Training-(Train Test Splet) Preperation

```
# create two data frames to store the article and the labels values
x_df = df['article']
y_df = df['label']
```

```
# Initializae count vectorizer
count_vectorizer = CountVectorizer()
# Fitting the articles' dataframe into term-document matrix
count_vectorizer.fit_transform(x_df)
# Transform the term-document matrix into term frequency matrix
freq_term_matrix = count_vectorizer.transform(x_df)
# normalization
tfidf = TfidfTransformer(norm = "l2")
# transform term frequency matrix to TF-IDF matrix
tfidf.fit(freq_term_matrix)
tf_idf_matrix = tfidf.fit_transform(freq_term_matrix)
```

```
# Show the TF-IDF Matrix
print(tf_idf_matrix)
```

```
  (0, 67464)    0.02973812519296478
  (0, 67321)    0.02187381783762338
  (0, 67058)    0.03670490726111059
  (0, 67047)    0.02497991051631268
  (0, 67003)    0.03608737631663893
  (0, 66960)    0.029638975508019683
  (0, 66957)    0.008494407784163263
  (0, 66837)    0.020235752291433554
  (0, 66758)    0.01628131226071343
  (0, 66660)    0.027665603425261122
  (0, 66657)    0.016397372174331758
  (0, 66595)    0.04086614079941994
  (0, 66507)    0.02497991051631268
  (0, 66362)    0.03427952381517232
  (0, 66278)    0.034909727605850045
  (0, 66252)    0.01794639877643502
  (0, 66140)    0.013574364289387776
  (0, 66103)    0.06063351246956065
  (0, 65987)    0.011740783232089389
  (0, 65942)    0.02653965444827103
  (0, 65833)    0.041980947334796015
  (0, 65750)    0.03370758797740281
  (0, 65685)    0.0230449265457275
  (0, 65584)    0.022318445242516183
  (0, 65581)    0.04243337444452707
  :     :
  (6334, 5639)  0.038250323170883435
  (6334, 5332)  0.0354062936721994
  (6334, 5313)  0.09028971654667797
  (6334, 5311)  0.09798440639750423
  (6334, 5301)  0.02195322850832939
  (6334, 4741)  0.017747857962236085
  (6334, 4653)  0.03335352189913419
  (6334, 4606)  0.033277569041059336
  (6334, 4395)  0.05425812802894449
  (6334, 4330)  0.03936095812786309
  (6334, 4302)  0.02605290031231262
  (6334, 4299)  0.026789570588873202
  (6334, 4298)  0.035610232881598286
  (6334, 4294)  0.029347965416418473
  (6334, 4175)  0.035158984694039726
  (6334, 3830)  0.03141535749456796
  (6334, 3744)  0.03998531447816947
  (6334, 3360)  0.023548245790150864
  (6334, 3225)  0.0331279496213474
  (6334, 2785)  0.029085232887758293
  (6334, 2462)  0.03919178458421316
  (6334, 2011)  0.01680217194235966
  (6334, 1248)  0.0586070551598393
  (6334, 780)   0.027913295574296025
  (6334, 614)   0.015249421621680597
```

```
# Splitting the data into test data and train data
x_train, x_test, y_train, y_test = train_test_split(tf_idf_matrix,y_df, random_state=0)
```

```
x_train.shape , y_train.shape
```

```
((4751, 68276), (4751,))
```

```
x_test.shape , y_test.shape
```

```
((1584, 68276), (1584,))
```

5.2 Linear Support Vector Classification

```
# Initializing linear SVC classifier
ln_svc =  LinearSVC(class_weight='balanced')

# Training the classifier model on the training data
%time ln_svc.fit(x_train, y_train)

# Making predection on the training and testing data
y_pred_train = ln_svc.predict(x_train)
y_pred_test = ln_svc.predict(x_test)

# calculating and prenting the accuracy score
print("\nTraining Accuracy score:",accuracy_score(y_train, y_pred_train))
print("Testing Accuracy score:",accuracy_score(y_test, y_pred_test))
```

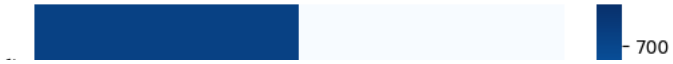
```
CPU times: user 274 ms, sys: 0 ns, total: 274 ms
Wall time: 290 ms

Training Accuracy score: 0.9997895179962113
Testing Accuracy score: 0.9324494949494949
```

The SVC model completed an exceptionally high training accuracy with approximately 99% after few repetitions of training. Also, the SVC testing accuracy is around 93.24%. This shows that the model effectively classifies the pursued real or fake news articles, proving its effectiveness on new or unknown data.

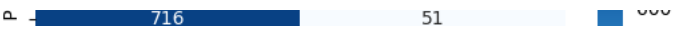
6.1 Linear Support Vector Evaluation

6.1.1 SVC Confusion Matrix

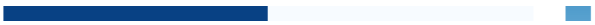
```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred_test)

# Create dataframe to display the confusion matrix
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive', 'Actual Negative'],
                         index=['Predict Positive', 'Predict Negative'])

# Generate and Display the confusion matrix
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='Blues')
plt.show()
```



The confusion matrix indicates the high accuracy in predection of the True Positice and True Negative done by the model.



6.1.2 SVC Classification Report



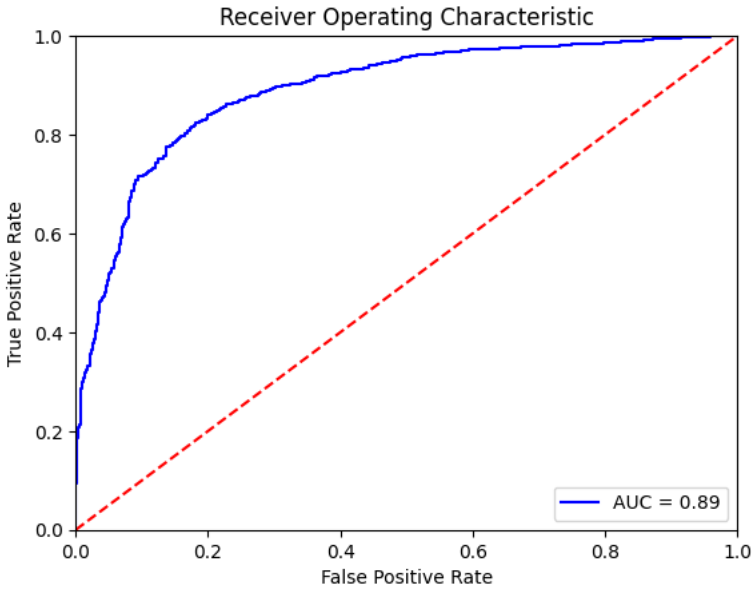
```
# print the classification report metrics for the SVC model
print(classification_report(y_test, y_pred_test, target_names=['Fake', 'Real']))
```

```
              precision    recall  f1-score   support

        Fake       0.81      0.82      0.82       767
        Real       0.83      0.82      0.83       817

    accuracy                           0.82      1584
   macro avg       0.82      0.82      0.82      1584
weighted avg       0.82      0.82      0.82      1584
```

The Classification report of the SVC Classifier shows that the model has preformed well in diffrentiating between the "Fake" and "Real" news, with precision, recall and F1 score of 82%.

6.1.3 SVC AUC-ROC

```
# Calculating the probability of the text in the dataset
probs = svc._predict_proba_lr(x_test)
preds = probs[:,1]

# Calculating fase positive rate and true positive rate
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

# Plot the curve
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



The AUC- ROC indicates the high discriminatory ability of the model in distanquishing between "FAKE" and "REAL" news.

5.3 Logistic Regression Classification Model

```
# Initializing LogisticRegression classifier
lr = LogisticRegression()

# Training the classifier model on the training data
%time lr.fit(x_train, y_train)

# Making predection on the training and testing data
y_pred_train = lr.predict(x_train)
y_pred_test = lr.predict(x_test)

# calculating and prenting the accuracy score
print("\nTraining Accuracy score:",accuracy_score(y_train, y_pred_train))
print("Testing Accuracy score:",accuracy_score(y_test, y_pred_test))
```

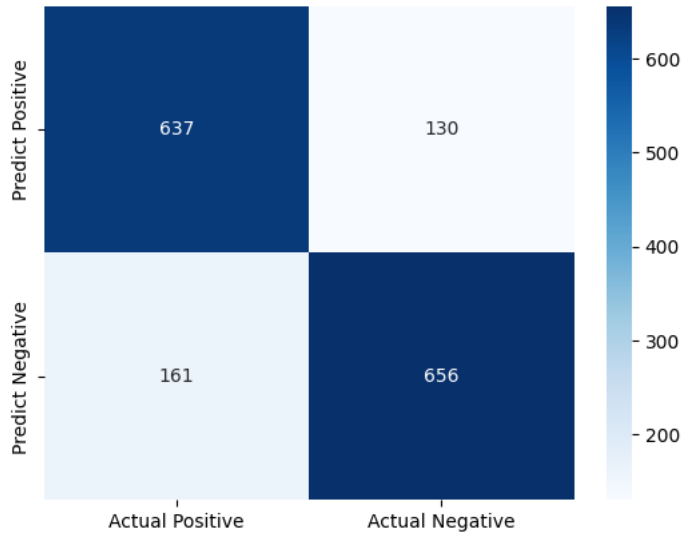
```
CPU times: user 777 ms, sys: 248 ms, total: 1.02 s
Wall time: 1.36 s

Training Accuracy score: 0.9231740686171332
Testing Accuracy score: 0.8162878787878788
```

The training accuracy of the LG model was around 92.32%. This indicates that the model did greatly with the training set of data. However, the testing accuracy is lower than SVC model where it scored only 81.62%

6.2 Logistic Regression Evaluation

6.2.1 Logistic Regression Confusion Matrix

```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred_test)

# Create dataframe to display the confusion matrix
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive', 'Actual Negative'],
                         index=['Predict Positive', 'Predict Negative'])

# Generate and Display the confusion matrix
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='Blues')
plt.show()
```



6.2.2 Logistic Regression Classification Report

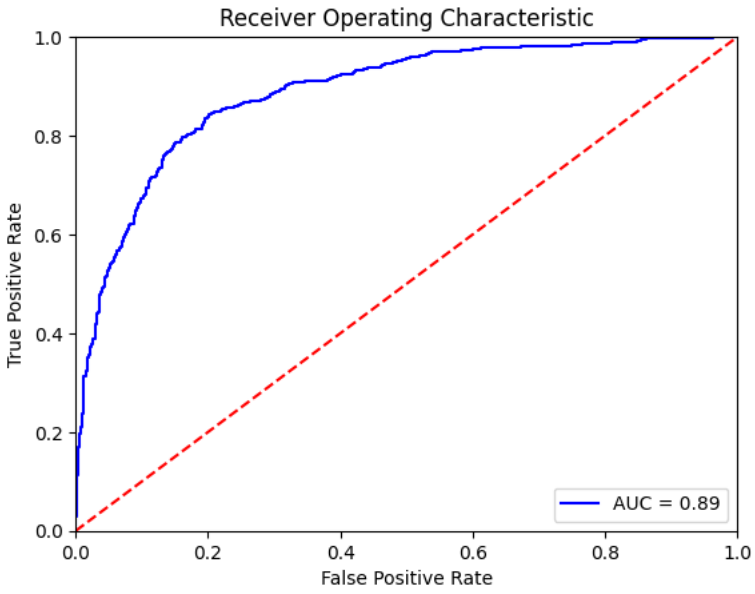
```
# print the classification report metrics for Logistic Regression classification model
print(classification_report(y_test, y_pred_test, target_names=['Fake', 'Real']))
```

```
              precision    recall  f1-score   support

        Fake       0.80      0.83      0.81       767
        Real       0.83      0.80      0.82       817

    accuracy                           0.82      1584
   macro avg       0.82      0.82      0.82      1584
weighted avg       0.82      0.82      0.82      1584
```

The Classification report of the Logistic Regression Classifier shows that the model has preformed well in diffrentiating between the "Fake" and "Real" news, with precision, recall and F1 score of 82%.

6.2.3 Logistic Regression AUC-ROC

```
probs = lr.predict_proba(x_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



The AUC- ROC indicates the high discriminatory ability of the model in distanquishing between "FAKE" and "REAL" news.

Conclusion

In conclusion, to answer the question asked in the beginning whether if machine learning can help with accurately detect fake news, NLP techniques paired with machine learning models like Logistic Regression and Linear Support Vector Classification show potential in the detection of fake news. In addition to all features engineering like POS tagging, NER, and TF-IDF which are essential to the performance of the chosen model. The SVC and LG classifiers both performed equitably, but the Support Vector Classification model performed slightly better in identifying fake news using the dataset that was supplied. With the help of NLP and machine learning approaches, this project provides a foundation for future research and development in the field of fake news detection. for future recommendation, news classification should be

done encountering various of ML algorithms such as Naive Bayesm Random Forest and Gradient Boosing with hyperparameters of the selected models to improve performance.

Project Log

NLP Project Preference Document				IT9002
Mentor : Sini Raj Pulari				
Project Name:	NLP			
Student Name:	Maryam Shaabam			Student I ID: 202306805
NLP Project Log File				
Date	Week No	Task Name	Work done During the week (Bullet points /Description)	Issues Experienced if any
11-Nov-23	1	Task1: Problem Statement Formulation and definition	evaluating project topics searching for datasets in order to formulate and define problem statment	trying define the problem statment without exploring the topic or datasets required for the project.
18-Nov-23	2	Task 2: Selection of an appropriate data set	Searching for an appropriate data set	while searching for an appropriate data set for the chosen topic (plagiarisim detection) I faced diffeculties finding useful datasets. Therefore I had to change the topic to fake news detection in in order to proceed with the tasks
25-Nov-23	3	Task 3: Text preprossesing	testing text processing methods on selected datasets.	time management
2-Dec-23	4	Task 4: Text representation	Still working on text representation	
11-Dec-23	5	Task 5: Text Classification	Completed Text Representation and SVC Classifier. Still working on the other classifier.	

Start coding or generate with AI.

GitHub Link

https://github.com/Memz214/NLP-project.git

+ Code + Text

Refrences

O'Brien. (2018, June). Machine Learning for Detection of Fake News. Massachusetts Institute of Technology. Retrieved December 1, 2023, from https://dspace.mit.edu/bitstream/handle/1721.1/119727/1078649610-MIT.pdf

Ahmed, Hinkelmann, & Corradini . (2020). Development of Fake News Model Using Machine Learning through Natural Language Processing . In Cornell University. Retrieved December 7, 2023, from https://arxiv.org/ftp/arxiv/papers/2201/2201.07489.pdf#:~:text=Support%20Vector%20Machine%20is%20one,news%20detection%20problem%20%5B20%5D.&text=classification%20%5B27%5D.,like%20Online%20Perceptron%20and%20MIRA.

Bedi, G. (2020, July 13). A guide to Text Classification(NLP) using SVM and Naive Bayes with Python. Medium. https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34

Richards, S., & Richards, S. (2022, July 11). Machine Learning (ML) for Natural Language Processing (NLP) - Lexalytics. Lexalytics. https://www.lexalytics.com/blog/machine-learning-natural-language-processing/