Surya Group of Institutons

Naan Mudhalvan

IBM - Artificial Integllience

MANIKANDAN T

422221104020

Team - 7

Fake News Detection Using NLP Machine Learning Algorithm

Exploring the fake news dataset, performing data analysis such as word clouds and ngrams, and fine-tuning BERT transformer to build a fake news detector in Python using transformers library.

Fake news is the intentional broadcasting of false or misleading claims as news, where the statements are purposely deceitful.

Newspapers, tabloids, and magazines have been supplanted by digital news platforms, blogs, social media feeds, and a plethora of mobile news applications. News organizations benefitted from the increased use of social media and mobile platforms by providing subscribers with up-to-the-minute information.

Consumers now have instant access to the latest news. These digital media platforms have increased in prominence due to their easy connectedness to the rest of the world and allow users to discuss and share ideas and debate topics such as democracy, education, health, research, and history. Fake news items on digital

platforms are getting more popular and are used for profit, such as political and financial gain.

- Introduction
- How Big is this Problem?
- The Solution
- Data Exploration
    - Distribution of Classes
- Data Cleaning for Analysis
- Explorative Data Analysis
    - Single-word Cloud
    - Most Frequent Bigram (Two-word Combination)
    - Most Frequent Trigram (Three-word Combination)
- Building a Classifier by Fine-tuning BERT
    - Data Preparation
    - Tokenizing the Dataset
        - Loading and Fine-tuning the Model
        - Model Evaluation

- Appendix: Creating a Submission File for Kaggle
- Conclusion

Data Exploration:

we utilized the fake news dataset from Kaggle to classify untrustworthy news articles as fake news. We have a complete training dataset containing the following characteristics:

- id: unique id for a news article
- title: title of a news article
- author: author of the news article
- text: text of the article; could be incomplete
- label: a label that marks the article as potentially unreliable denoted by 1 (unreliable or fake) or 0 (reliable).

It is a binary classification problem in which we must predict if a particular news story is reliable or not.

If you have a Kaggle account, you can simply download the dataset from the website there and extract the ZIP file.

I also uploaded the dataset into Google Drive, and you can get it here or use the gdown library to download it in Google Colab or Jupyter notebooks automatically:

$ pip install gdown

download from Google Drive

$ gdown "https://drive.google.com/uc?id=178f_VkNxccNidap-5-uffXUW475pAuPy&confirm=t"

Downloading...

From: https://drive.google.com/uc?id=178f_VkNxccNidap-5-uffXUW475pAuPy&confirm=t

To: /content/fake-news.zip

100% 48.7M/48.7M [00:00<00:00, 74.6MB/s]

Unzipping the files:

$ unzip fake-news.zip

Installing the required dependencies:

$ pip install transformers nltk pandas numpy matplotlib seaborn wordcloud

Let's import the essential libraries for analysis:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

The NLTK corpora and modules must be installed using the standard NLTK downloader:

```
import nltk

nltk.download('stopwords')

nltk.download('wordnet')
```

The fake news dataset comprises various authors' original and fictitious article titles and text. Let's import our dataset:

```
# load the dataset

news_d = pd.read_csv("train.csv")

print("Shape of News data:", news_d.shape)

print("News data columns", news_d.columns)
```

Output:

```
Shape of News data: (20800, 5)
```

News data columns Index(['id', 'title', 'author', 'text', 'label'], dtype='object')

Here's how the dataset looks:

 # by using df.head(), we can immediately familiarize ourselves with the dataset.

news_d.head()

Output:

id title author text label

0 0 House Dem Aide: We Didn't Even See Comey's Let... Darrell Lucus House Dem Aide: We Didn't Even See Comey's Let... 1

1 1 FLYNN: Hillary Clinton, Big Woman on Campus - ... Daniel J. Flynn Ever get the feeling your life circles the rou... 0

2 2 Why the Truth Might Get You Fired Consortiumnews.com Why the Truth Might Get You Fired October 29, ... 1

3 3 15 Civilians Killed In Single US Airstrike Hav... Jessica Purkiss Videos 15 Civilians Killed In Single US Airstr... 1

4 4 Iranian woman jailed for fictional unpublished... Howard Portnoy Print \nAn Iranian woman has been sentenced to... 1

We have 20,800 rows, which have five columns. Let's see some statistics of the text column:

#Text Word startistics: min.mean, max and interquartile range

txt_length = news_d.text.str.split().str.len()

txt_length.describe()

Output:

count 20761.000000

mean 760.308126

std 869.525988

min 0.000000

25% 269.000000

50% 556.000000

75% 1052.000000

max 24234.000000

Name: text, dtype: float64

Stats for the title column:

#Title statistics

title_length = news_d.title.str.split().str.len()

title_length.describe()

Output:

count 20242.000000

mean 12.420709

std 4.098735

min 1.000000

25% 10.000000

50% 13.000000

75% 15.000000

max 72.000000

Name: title, dtype: float64

The statistics for the training and testing sets are as follows:

- **The text attribute has a higher word count with an average of 760 words and 75% having more than 1000 words.**
- **The title attribute is a short statement with an average of 12 words, and 75% of them are around 15 words.**

Distribution of Classes:

Counting plots for both labels:

```
sns.countplot(x="label", data=news_d);

print("1: Unreliable")

print("0: Reliable")

print("Distribution of labels:")

print(news_d.label.value_counts());
```

Output:

1: Unreliable

0: Reliable

Distribution of labels:

1 10413

0 10387

Name: label, dtype: int64

print(round(news_d.label.value_counts(normalize=True),2)*100);

Output:

1 50.0

0 50.0

Name: label, dtype: float64

Data Cleaning for Analysis :

- **Drop unused rows and columns.**
- **Perform null value imputation.**
- **Remove special characters.**
- **Remove stop words.**

 # Constants that are used to sanitize the datasets

column_n = ['id', 'title', 'author', 'text', 'label']

remove_c = ['id','author']

```
categorical_features = []

target_col = ['label']

text_f = ['title', 'text']


# Clean Datasets

import nltk

from nltk.corpus import stopwords

import re

from nltk.stem.porter import PorterStemmer

from collections import Counter


ps = PorterStemmer()

wnl = nltk.stem.WordNetLemmatizer()


stop_words = stopwords.words('english')

stopwords_dict = Counter(stop_words)


# Removed unused clumns

def remove_unused_c(df,column_n=remove_c):
```

```
    df = df.drop(column_n,axis=1)
    return df


# Impute null values with None
def null_process(feature_df):
    for col in text_f:
        feature_df.loc[feature_df[col].isnull(), col] = "None"
    return feature_df


def clean_dataset(df):
    # remove unused column
    df = remove_unused_c(df)
    #impute null values
    df = null_process(df)
    return df


# Cleaning text from unused characters
def clean_text(text):
    text = str(text).replace(r'http[¥w:/¥.]+', ' ')    # removing urls
```

```
    text = str(text).replace(r'[^¥.¥w¥s]', ' ')    # remove everything but characters and punctuation

    text = str(text).replace('[^a-zA-Z]', ' ')

    text = str(text).replace(r'¥s¥s+', ' ')

    text = text.lower().strip()

    #text = ' '.join(text)

    return text


## Nltk Preprocessing include:

# Stop words, Stemming and Lemmetization

# For our project we use only Stop word removal

def nltk_preprocess(text):

    text = clean_text(text)

    wordlist = re.sub(r'[^¥w¥s]', '', text).split()

    #text = ' '.join([word for word in wordlist if word not in stopwords_dict])

    #text = [ps.stem(word) for word in wordlist if not word in stopwords_dict]

    text = ' '.join([wnl.lemmatize(word) for word in wordlist if word not in stopwords_dict])

    return   text
```

In the code block above:

- **We have imported NLTK, which is a famous platform for developing Python applications that interact with human language. Next, we import re for regex.**
- **We import stopwords from nltk.corpus. When working with words, particularly when considering semantics, we sometimes need to eliminate common words that do not add any significant meaning to a statement, such as "but", "can", "we", etc.**
- **PorterStemmer is used to perform stemming words with NLTK. Stemmers strip words of their morphological affixes, leaving the word stem solely.**
- **We import WordNetLemmatizer() from NLTK library for lemmatization. Lemmatization is much more effective than stemming. It goes beyond word reduction and evaluates a language's whole lexicon to apply morphological analysis to words, with the goal of just removing inflectional ends and returning the base or dictionary form of a word, known as the lemma.**
- **stopwords.words('english') allow us to look at the list of all the English stop words supported by NLTK.**
- **remove_unused_c() function is used to remove the unused columns.**
- **We impute null values with None using the null_process() function.**
- **Inside the function clean_dataset(), we call remove_unused_c() and null_process() functions. This function is responsible for data cleaning.**

- **To clean text from unused characters, we have created the clean_text() function.**
- **For preprocessing, we will use only stop word removal. We created the nltk_preprocess() function for that purpose.**