

Fake News Detection using NLP

**Abstract**

The spreading of fake news has given rise to many problems in society. It is due to its ability to cause a lot of social and national damage with destructive impacts. Sometimes it gets very difficult to know if the news is genuine or fake. Therefore it is very important to detect if the news is fake or not. Fake News is a term used to represent fabricated news or propaganda comprising misinformation communicated through traditional media channels like print, and television as well as non-traditional media channels like social media. Techniques of NLP and Machine learning can be used to create models which can help to detect fake news.

**Introduction**

Since a lot of time is spent by users on social media and people prefer online means of information it has become difficult to know about the authenticity of the news. People acquire most of the information by these means as it is free and can be accessed from anywhere irrespective of place and time. Since this data can be put out by anyone there is lack of accountability in it which makes it less trustable unlike the traditional methods of gaining information like newspapers or some trusted source. Fake news is dangerous as it can deceive people easily and create a state of confusion among a community. This can further affect the society badly .The spread of fake news creates rumours circulating around and the victims could be badly impacted.

**Problem Statement**

The problem at hand is the development of an automated system capable of classifying news articles into two distinct categories: "fake" and "real." In an era of information overload, the spread of misinformation, and the potential consequences it entails, there is a critical need for tools that can help identify and differentiate between news content that is factually accurate and trustworthy ("real") and content that is misleading, fabricated, or false ("fake").

The project's primary goal is to build a machine learning model that can accurately distinguish between genuine news articles and fake news articles. Achieving this would be a significant step toward enhancing media literacy and trust in news sources.

**Design Thinking**

1. **Understanding the Problem**: We started by gaining a deep understanding of the problem of misinformation in news and its consequences.
2. **Data Collection:** We obtained the dataset from Kaggle, consisting of labeled news articles.

***Dataset Link*:** <https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset>

1. **Data Preprocessing:** We cleaned and preprocessed the text data, handled missing values, and converted it into a suitable format for machine learning.
2. **Feature Extraction:** We used text vectorization techniques, such as TF-IDF, to represent the text data as numerical features.
3. **Model Selection:** We experimented with several classification algorithms to choose the best-performing one.
4. **Model Training**: We trained the selected model on the preprocessed data.
5. **Evaluation:** We assessed the model's performance using appropriate metrics, such as accuracy and precision.

**Phase of Development**

Firstly, the datasets are collected. The datasets are then merged to obtain a master dataset.

This dataset is then preprocessed. Preprocessing of the datasets include lowering of the data, stop word removal, stemming, tokenization and padding is also performed in order to obtain the same length.

1. ***Tokenization****:* Tokenization is the process of breaking down a stream of text into tokens, which can be words, phrases, symbols, or any other significant items. This step's major purpose is to extract individual words in a sentence. The tokenization is done on each text in the dataset.
2. ***Stop Words****:* Stop words are the commonly used words and are removed from the text as they do not add any value to the analysis. These phrases have little or no meaning. A list of terms that are regarded as stop words in the English language is included in the NLTK library. All the stop words from the texts are removed.
3. ***Capitalization****:* Sentences can have a combination of capital and lowercase letters. A written document is made up of multiple sentences. One of the method for reducing the issue space is to convert everything to lower case. This aligns all of the words in a document in the same location. Using the python function, all the words are converted to lower case.
4. ***Stemming****:* Stemming is the process of reducing the words to its root form by eliminating extraneous characters. Porter Stemmer is one of the stemming model which is used here to convert the words into its root form.
5. ***Lemmatization:*** Text lemmatization is the process of removing a word's superfluous prefix or suffix and extracting the basic word. All the suffixes and prefixes from the words are removed to reduce space.

The dataset is then split into training data and testing data. To overcome the problem of detecting fake news this project proposes 6 similar LSTM models which are to be trained and each model will be fed with the different text vectors of news headline and news content. This will help in obtaining a good model which will tell if the news is true or it is fake. In this project we have used six similar LSTM models.

Three text vectorization techniques are used which are GloVe, Word2vec and TF-IDF. The first LSTM model will be fed with the vectors of the title of the news using GloVe. The second model will be fed with the vectors of the content of the news using GloVe. Similarly, two models will be built using the Word2vec technique each for the title of the news and the content of the news respectively.

Lastly, the LSTM model will be fed with the text vectors of the title of the news using TF-IDF and another model will be fed with the text vectors of the content of the news using TF-IDF. By doing so we can identify which technique gives better results and identify which model performs well. Lastly, the performance is measured using the performance metrics accuracy, precision and recall.

The libraries used are

* Pandas: For importing the dataset.
* Seaborn /Matplotlib : For data visualization.
* NLTK(Natural language toolkit): For preprocessing

Code:

#Import Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import nltk

import re

import string

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

import keras

from keras.preprocessing import text,sequence

from keras.models import Sequential

from keras.layers import Dense,Embedding,LSTM,Dropout

import warnings

warnings.filterwarnings('ignore')

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

/kaggle/input/fake-and-real-news-dataset/True.csv

/kaggle/input/fake-and-real-news-dataset/Fake.csv

#Load and Check Data

real\_data = pd.read\_csv('/kaggle/input/fake-and-real-news-dataset/True.csv')

fake\_data = pd.read\_csv('/kaggle/input/fake-and-real-news-dataset/Fake.csv')

data = pd.concat([real\_data, fake\_data], ignore\_index=True, sort=False)

#Visualization

// Count of Fake and Real Data

print(data["target"].value\_counts())

fig, ax = plt.subplots(1,2, figsize=(19, 5))

g1 = sns.countplot(data.target,ax=ax[0],palette="pastel");

g1.set\_title("Count of real and fake data")

g1.set\_ylabel("Count")

g1.set\_xlabel("Target")

g2=plt.pie(data["target"].value\_counts().values,explode=[0,0],

labels=data.target.value\_counts().index,autopct='%1.1f%%',

colors=['SkyBlue','PeachPuff'])

fig.show()

//Distribution of The Subject According to Real and Fake Data

print(data.subject.value\_counts())

plt.figure(figsize=(10, 5))

ax = sns.countplot(x="subject", hue='target', data=data, palette="pastel")

plt.title("Distribution of The Subject According to Real and Fake Data")

#Data Cleaning

data['text']= data['subject'] + " " + data['title'] + " " + data['text']

// Removal of HTML Contents

from bs4 import BeautifulSoup

soup = BeautifulSoup(first\_text, "html.parser")

//Removal of Punctuation Marks and Special Characters

first\_text = soup.get\_text()

first\_text = re.sub('\[[^]]\*\]', ' ', first\_text)

first\_text = re.sub('[^a-zA-Z]',' ',first\_text) # replaces non-alphabets with spaces

first\_text = first\_text.lower() # Converting from uppercase to lowercase

// Removal of Stopwords

nltk.download("stopwords")

from nltk.corpus import stopwords

# we can use tokenizer instead of split

first\_text = nltk.word\_tokenize(first\_text)

first\_text = [ word for word in first\_text if not word in set(stopwords.words("english"))]

//Lemmatization

lemma = nltk.WordNetLemmatizer()

first\_text = [ lemma.lemmatize(word) for word in first\_text]

first\_text = " ".join(first\_text)

#Removal of HTML Contents

def remove\_html(text):

soup = BeautifulSoup(text, "html.parser")

return soup.get\_text()

#Removal of Punctuation Marks

def remove\_punctuations(text):

return re.sub('\[[^]]\*\]', '', text)

# Removal of Special Characters

def remove\_characters(text):

return re.sub("[^a-zA-Z]"," ",text)

#Removal of stopwords

def remove\_stopwords\_and\_lemmatization(text):

final\_text = []

text = text.lower()

text = nltk.word\_tokenize(text)

for word in text:

if word not in set(stopwords.words('english')):

lemma = nltk.WordNetLemmatizer()

word = lemma.lemmatize(word)

final\_text.append(word)

return " ".join(final\_text)

#Total function

def cleaning(text):

text = remove\_html(text)

text = remove\_punctuations(text)

text = remove\_characters(text)

text = remove\_stopwords\_and\_lemmatization(text)

return text

#Apply function on text column

data['text']=data['text'].apply(cleaning)

//WordCloud for Real News

from wordcloud import WordCloud,STOPWORDS

plt.figure(figsize = (15,15))

wc = WordCloud(max\_words = 500 , width = 1000 , height = 500 ,

stopwords = STOPWORDS).generate(" ".join(data[data.target == 1].text))

plt.imshow(wc , interpolation = 'bilinear')

plt.figure(figsize = (15,15))

//WordCloud for Fake News

wc = WordCloud(max\_words = 500 , width = 1000 , height = 500 ,

stopwords = STOPWORDS).generate(" ".join(data[data.target == 0].text))

plt.imshow(wc , interpolation = 'bilinear')

//Number of words in each text

fig,(ax1,ax2)=plt.subplots(1,2,figsize=(12,8))

text\_len=data[data['target']==0]['text'].str.split().map(lambda x: len(x))

ax1.hist(text\_len,color='SkyBlue')

ax1.set\_title('Fake news text')

text\_len=data[data['target']==1]['text'].str.split().map(lambda x: len(x))

ax2.hist(text\_len,color='PeachPuff')

ax2.set\_title('Real news text')

fig.suptitle('Words in texts')

plt.show()

texts = ' '.join(data['text'])

string = texts.split(" ")

def draw\_n\_gram(string,i):

n\_gram = (pd.Series(nltk.ngrams(string, i)).value\_counts())[:15]

n\_gram\_df=pd.DataFrame(n\_gram)

n\_gram\_df = n\_gram\_df.reset\_index()

n\_gram\_df = n\_gram\_df.rename(columns={"index": "word", 0: "count"})

print(n\_gram\_df.head())

plt.figure(figsize = (16,9))

return sns.barplot(x='count',y='word', data=n\_gram\_df)

#Modeling

//Train Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data['text'], data['target'], random\_state=0)

//Tokenizing

max\_features = 10000

maxlen = 300

tokenizer = text.Tokenizer(num\_words=max\_features)

tokenizer.fit\_on\_texts(X\_train)

tokenized\_train = tokenizer.texts\_to\_sequences(X\_train)

X\_train = sequence.pad\_sequences(tokenized\_train, maxlen=maxlen)

tokenized\_test = tokenizer.texts\_to\_sequences(X\_test)

X\_test = sequence.pad\_sequences(tokenized\_test, maxlen=maxlen)

//Training LSTM Model

batch\_size = 256

epochs = 10

embed\_size = 100

model = Sequential()

#Non-trainable embeddidng layer

model.add(Embedding(max\_features, output\_dim=embed\_size, input\_length=maxlen, trainable=False))

#LSTM

model.add(LSTM(units=128 , return\_sequences = True , recurrent\_dropout = 0.25 , dropout = 0.25))

model.add(LSTM(units=64 , recurrent\_dropout = 0.1 , dropout = 0.1))

model.add(Dense(units = 32 , activation = 'relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer=keras.optimizers.Adam(lr = 0.01), loss='binary\_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(X\_train, y\_train, validation\_split=0.3, epochs=10, batch\_size=batch\_size, shuffle=True, verbose = 1)

//Analysis After Training

print("Accuracy of the model on Training Data is - " , model.evaluate(X\_train,y\_train)[1]\*100 , "%")

print("Accuracy of the model on Testing Data is - " , model.evaluate(X\_test,y\_test)[1]\*100 , "%")

**Sources**

The open source platforms we use here are ;

* + Kaggle
  + Google colab
* Kaggle is for getting dataset and code we refer
* Google colab is for implementing the code

**Conclusion**

Fake news have increased in recent years and it has caused a lot of harm to the society. This project aimed to develop a model using the techniques of NLP and ML to detect if a news article/headline is fake or not and identify which methods give better output.

**Reference**

<https://www.kaggle.com/code/ilaydadu/fake-news-detection-with-nlp-and-lstm>

<https://www.kaggle.com/code/ilyapozdnyakov/fake-news-nlp-4-models-visualizations>