Create a Fake news detection using NLP

### By Hariharan.M-411421205014 (B.Tech/ Information Technology, 3rd year)

**Domain name: Artificial Intelligence**

**Project: To design a fake news detection innovation using NLP**

**Introduction to Fake News Detection Using NLP**

### In an era dominated by digital information, the rise of fake news poses a significant challenge to the credibility and reliability of news sources. False or misleading information, intentionally or unintentionally disseminated, has the potential to manipulate public opinion, influence decisions, and even disrupt societal harmony.

**NLP for fake news detection in step by step process :**

### Certainly, here's a step-by-step process for implementing Natural Language Processing (NLP) in the context of fake news detection:

1. **Data Collection:**

Collect a diverse dataset containing labeled examples of both real and fake news articles, tweets, or other textual sources.

1. **Text Preprocessing:**

Perform data cleaning by removing HTML tags, special characters, and irrelevant symbols. Tokenize the text into individual words or sub-word unit. Apply lemmatization or stemming to reduce words to their base or root for Remove stop words to focus on meaningful content.

1. **Text Representation:**

Choose a text representation method such as TF-IDF, Word2Vec, GloVe, or BERT embeddings.

Transform the textual data into numerical vectors that machine learning models can process.

1. **Feature Extraction:**

Extract additional features, such as sentiment analysis scores, part-of- speech tags, or named entities, to provide the model with more context.

1. **Model Selection:**

Choose a suitable machine learning or deep learning model for classification (e.g., Logistic Regression, Naive Bayes, Random Forest, LSTM, BERT-based models).

1. **Train-Test Split:**

Split the dataset into training and testing sets to evaluate the model's performance.

1. **Model Training:**

Train the selected model on the training set using the features extracted from the preprocessed text.

1. **Evaluation Metrics:**

Evaluate the model's performance on the testing set using metrics like accuracy, precision, recall, F1-score, and area under the ROC curve.

1. **Ensemble Learning (Optional):**

Implement ensemble methods to combine predictions from multiple models for improved accuracy.

1. **Explainability and Interpretability:**

* Incorporate methods for explaining model predictions (e.g., LIME, SHAP) to enhance transparency and interpretability.

1. **User Feedback Integration:**

* Allow users to provide feedback on news classifications and use this feedback for continuous model improvement.

1. **Bias Mitigation:**

* Identify and mitigate biases in the data and model to ensure fair and unbiased results.

1. **Real-time Monitoring:**

* Develop a system for real-time monitoring of news sources and updates to adapt quickly to emerging trends and challenges.

1. **Collaboration with Fact-Checking Organizations:**

* Collaborate with external fact-checking organizations to enhance the model's credibility and accuracy.

1. **Continuous Learning:**

* Implement mechanisms for the model to learn and adapt over time, considering evolving language patterns and emerging types of fake news

By following these steps, you can build a robust fake news detection system using NLP, combining advanced techniques with model interpretability

**Import Libraries**

Creating a library for fake news detection involves using natural language processing (NLP) and machine learning (ML) techniques. Popular tools include NLTK, spacy, and scikit-learn in Python. For deep learning, TensorFlow and PyTorch are common. Datasets like Fake News Dataset or LIAR-PLUS can be used for training. Implementing algorithms such as TF-IDF, Word Embeddings, or models like LSTM or BERT can enhance accuracy. Regularly updating the library with new data is crucial for effectiveness.

In [1]:

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

Let's assume you have a CSV file containing your fake news dataset. Here's a basic example of how you can load and check the data using Python:

import pandas as pd

# Adjust the file path accordingly

file\_path = 'path/to/your/dataset.csv'

# Load the dataset into a pandas DataFrame

df = pd.read\_csv(file\_path)

# Display the first few rows of the dataset

print(df.head())

# Check basic statistics of the dataset

print(df.describe())

# Check for missing values

print(df.isnull().sum())

# Assuming 'text' is the column containing the news text and 'label' is the column with labels

X = df['text']

y = df['label']

Make sure to replace 'text' and 'label' with the actual column names from your dataset. If you encounter missing values or other issues during this process, you might need to perform additional data cleaning steps based on the characteristics of your dataset.

In [2]:

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')

In [3]:

real\_data.head()

In [4]:

fake\_data.head()

# Visualization

Visualization is a powerful tool to understand and communicate patterns in data. For fake news detection, you can use various visualization techniques. Here's an example using Python with the matplotlib and seaborn libraries:

##### Count of Fake and Real Data

In [9]:

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()

0 23481

1 21417

Name: target, dtype: int64

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

In [10]:

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")

|  |  |  |
| --- | --- | --- |
| politicsNews | 11272 |  |
| worldnews | 10145 |  |
| News | 9050 |  |
| politics | 6841 |  |
| left-news | 4459 |  |
| Government News | 1570 |  |
| US\_News | 783 |  |
| Middle-east  Name: subject, | 778  dtype: int64 |  |

# Data Cleaning

Data cleaning is a crucial step in fake news detection to ensure the reliability and effectiveness of machine learning models. Here are some key aspects of data cleaning in this context:

**Text Cleaning:**

Remove HTML tags and special characters: Eliminate irrelevant symbols or HTML tags from the text.

Lowercasing: Convert all text to lowercase to ensure uniformity.

Remove stop words: Exclude common words (e.g., "and," "the") that don't contribute much to the analysis.

In [11]:

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

del data['subject'] del data['date'] data.head()

In [12]:

first\_text = data.text[10] first\_text

##### Handling Missing Data:

##### Identify and handle missing values appropriately, whether by imputation or removing instances with missing information.

##### Let's now remove everything except uppercase / lowercase letters using Regular Expressions.

In [15]:

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

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

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

first\_text

## Removal of Stopwords

##### Let's remove stopwords like is,a,the... Which do not offer much insight.

In [16]:

nltk.download("stopwords")

from nltk.corpus import stopwords

*# we can use tokenizer instead of split*

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

[nltk\_data] Downloading package stopwords to /usr/share/nltk\_data... [nltk\_data] Package stopwords is already up-to-date!

In [17]:

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

## Lemmatization

##### Lemmatization to bring back multiple forms of same word to their common root like 'coming', 'comes' into 'come'.

##### Lemmatization is a crucial step in natural language processing for tasks like fake news detection. It involves reducing words to their base or root form, which helps in standardizing the text data and improving the efficiency of language analysis. Here's how lemmatization contributes to fake news detection:

In [18]:

lemma = nltk.WordNetLemmatizer()

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

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

In [20]:

data.head()

**Let's make some visualization with new data.**

1. **WordCloud for Real News**

In [21]:

from wordcloud import WordCloud,STOPWORDS plt.figure(figsize = (15,15))

wc = WordCloud(max\_words = 500 , width = 1000 , height = 500 , stopwords = STOPWOR DS).generate(" ".join(data[data.target == 1].text))

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

#### 2.WordCloud for Fake News

In [22]:

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

wc = WordCloud(max\_words = 500 , width = 1000 , height = 500 , stopwords = STOPWOR DS).generate(" ".join(data[data.target == 0].text))

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

#### Number of words in each text

In [23]:

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()

##### The number of words seems to be a bit different. 500 words are most common in real news category while around 250 words are most common in fake news category.

In [24]:

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

In [25]:

string = texts.split(" ")

In [26]:

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)

## Unigram Analysis

In [27]:

draw\_n\_gram(string,1)

word count

0 (trump,) 149603

1 (said,) 133030

2 (u,) 78516

3 (state,) 62726

4 (president,) 58790

|  |  |  |
| --- | --- | --- |
| Bigram Analysis |  |  |
| In [28]:  draw\_n\_gram(string,2) |  |
| word  0 (donald, trump) | count 25203 |
| 1 (united, state) | 18943 |  |
| 2 (white, house) | 16296 |  |
| 3 (hillary, clinton) | 10217 |  |
| 4 (new, york) | 9305 |  |

## Trigram Analysis

In [29]:

draw\_n\_gram(string,3)

word count

1. (president, donald, trump) 6830
2. (pic, twitter, com) 6185
3. (featured, image, via) 6029
4. (president, barack, obama) 3911
5. (getty, image, news) 3575

## Modeling

Building effective models for fake news detection involves selecting appropriate machine learning algorithms, feature engineering, and careful model evaluation. Here are steps and considerations for modeling in fake news detection

Train Test Split

In [30]:

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

## Tokenizing

##### Tokenizing Text -> Representing each word by a number

* + Mapping of original word to number is preserved in word\_index property of tokenizer

Analysis After Training

In [38]:

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

print("Accuracy of the model on Testing Data is - " , model.evaluate(X\_test,y\_test

)[1]\*100 , "%")

1053/1053 [==============================] - 101s 96ms/step - loss: 0.0393 - a

ccuracy: 0.9843

Accuracy of the model on Training Data is - 98.42603802680969 %

351/351 [==============================] - 34s 97ms/step - loss: 0.0397 - accu

racy: 0.9840

Accuracy of the model on Testing Data is - 98.39643836021423 % In [39]:

plt.figure()

plt.plot(history.history["accuracy"], label = "Train") plt.plot(history.history["val\_accuracy"], label = "Test") plt.title("Accuracy")

plt.ylabel("Acc") plt.xlabel("epochs") plt.legend() plt.show()

In [40]:

plt.figure()

plt.plot(history.history["loss"], label = "Train") plt.plot(history.history["val\_loss"], label = "Test") plt.title("Loss")

plt.ylabel("Acc")

plt.xlabel("epochs") plt.legend() plt.show()

In [41]:

pred = model.predict\_classes(X\_test)

print(classification\_report(y\_test, pred, target\_names = ['Fake','Real']))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Fake | 1.00 | 0.97 | 0.98 | 5858 |
| Real | 0.97 | 1.00 | 0.98 | 5367 |
| accuracy |  |  | 0.98 | 11225 |
| macro avg | 0.98 | 0.98 | 0.98 | 11225 |
| weighted avg | 0.98 | 0.98 | 0.98 | 11225 |

## Conclusion

Detecting fake news is like putting together puzzle pieces. We use smart computer tools to understand the words and sentiments in news articles. It's a bit like teaching the computer to recognize when something doesn't sound quite right.

We also make sure our tools are like detectives, checking the details and sources to see if they match up. It's not a one-time job; it's like having a watchful eye that keeps learning and adapting to catch the tricky ways fake news can change. The goal is to create a system that helps us trust the news we read by making sure it's reliable and honest.