**FAKE NEWS DECTION USING NLP**

TEAM MEMBER

810021106030: HARINI V

**PHASE 2:INNOVATION**

**INTRODUCTION:**

Advanced techniques for enhancing fake news detection using Natural Language Processing (NLP) have become essential in our information-rich digital age. These techniques leverage the power of NLP and machine learning to effectively discern between truthful and deceptive information in textual content.

 Deep learning models like LSTM, BERT, and GPT have shown remarkable success in fake news detection.

**LSTM(LONG SHORT-TERM MEMORY):**

LSTM can capture temporal dependencies in textual data, which is beneficial for detecting fake news since the order of words can be critical.

You can preprocess news articles as sequences of words or embeddings and use LSTM to analyze them, considering the sequential nature of language.

**BERT(BIDERECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS):**

* BERT pre-trained models have a deep understanding of language context and semantics, making them effective for fake news detection.
* Fine-tuning BERT on a labeled dataset specific to fake news detection tasks can significantly improve accuracy.
* Utilize BERT’s ability to encode text into contextual embeddings and feed them to a classifier for prediction.
* Incorporating these techniques may involve natural language processing (NLP) libraries like TensorFlow or PyTorch and pre-trained models like GPT, BERT, or LSTM implementations in these libraries. It’s essential to preprocess your data, fine-tune the models, and evaluate their performance rigorously to achieve improved fake news detection accuracy.

**Data Preprocessing:**

* Collect a dataset of news articles, tweets, or any text data labeled as either real or fake news.
* **Tokenize the text:** Split the text into individual words or subword tokens.
* Create a vocabulary: Build a dictionary of unique words or tokens in your dataset.
* Convert text to numerical data: Map words/tokens to their corresponding numerical representations (word embeddings or one-hot encoding).

**Feature Engineering:**

* Advanced NLP techniques involve the creation of sophisticated linguistic features, such as sentiment analysis, emotion detection, and stance analysis.
* These features can reveal the emotional tone and stance of a news article, helping to assess its credibility

**Fake News Datasets:**

* The development of large, labeled fake news datasets is crucial for training and evaluating advanced models.
* These datasets enable researchers to fine-tune models specifically for fake news detection, improving their accuracy.

Real-time Monitoring:

* Continuous monitoring of news sources and social media platforms is essential to detect emerging fake news stories.

Advanced NLP techniques can automate this process and provide timely alerts.

**Fake news detection using BERT**

In[1]:

Import numpy as np

Import pandas as pd

Import matplotlib.pyplot as plt

Import os

For dirname, \_, filenames in os.walk(‘/kaggle/input’):

For filename in filenames:

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

Read Dataset

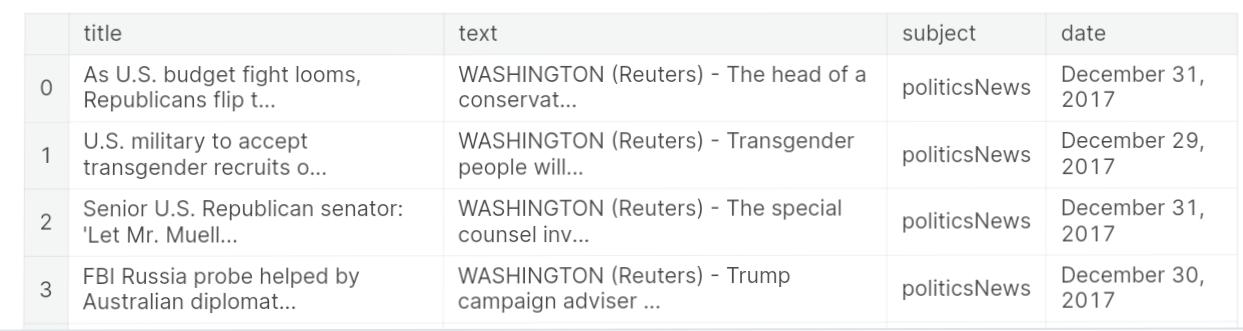
In[2]:

True\_news = pd.read\_csv(‘/kaggle/input/fake-and-real-news-dataset/True.csv’)

Fake\_news = pd.read\_csv(‘/kaggle/input/fake-and-real-news-dataset/Fake.csv’)

In[3]:

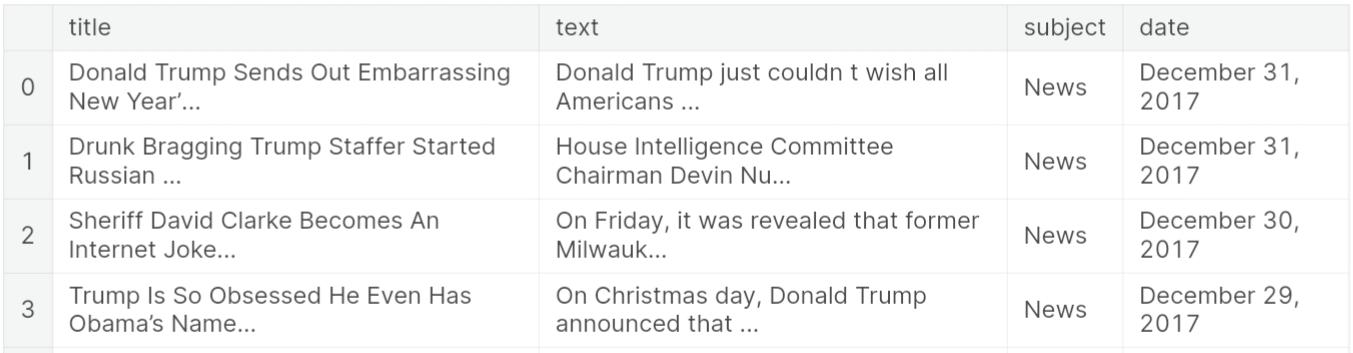
True\_news.head()

Out[3]:

In[4]:

Fake\_news.head()

Out[4]:



**Make new column with labels 1 for ‘true’, 0 for ‘false’.**

**In[5]:**

True\_news[‘target’] = [1]\*len(true\_news)

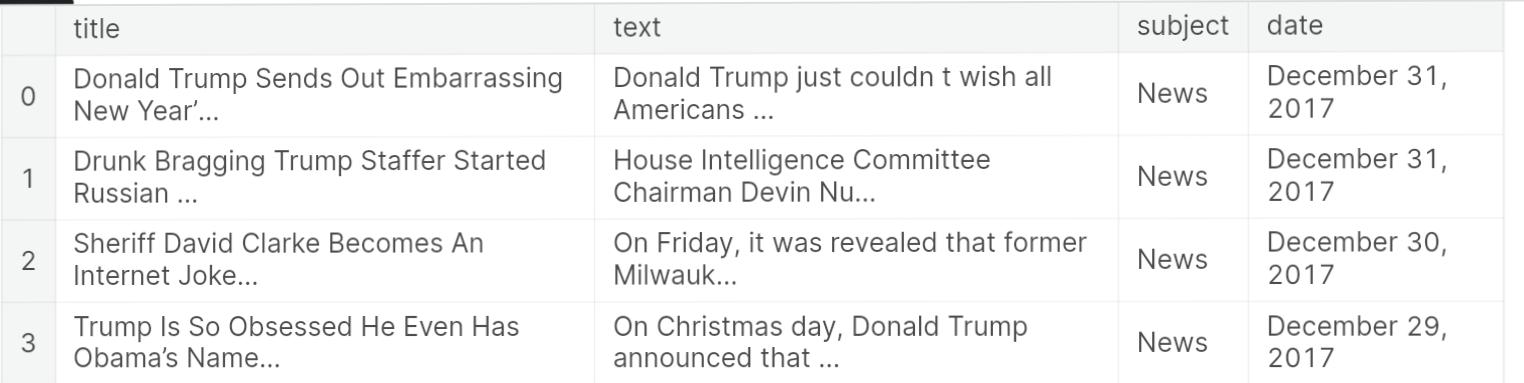
Fake\_news[‘target’] = [0]\*len(fake\_news)

**Append both the data to one dataframe.**

**In[6]:**

Dataset = true\_news.append(fake\_news).sample(frac = 1).reset\_index().drop(columns = [‘index’])

Dataset.head()

**Out[6]:**

**Visualize data to see the true/fake news ratio.**

**In[7]:**

Label\_size = [dataset[‘target’].sum(), len(dataset[‘target’])-dataset[‘target’].sum()]

Plt.pie(label\_size,explode=[0.1,0.1],colors=[‘firebrick’,’navy’],startangle=90,shadow=True,labels=[‘Fake’,’True’],autopct=’%1.1f%%’)

**Out[7]:**

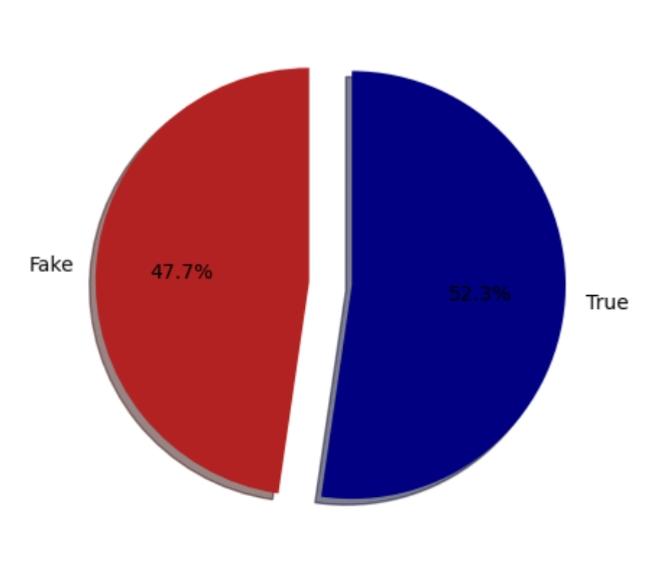
([<matplotlib.patches.Wedge at 0x7c202363e7d0>,

<matplotlib.patches.Wedge at 0x7c202363d090>],

[Text(-1.1968727148445069, 0.08657773651892332, ‘Fake’),

Text(1.1968727229504943, -0.08657762445961172, ‘True’)],

[Text(-0.6981757503259622, 0.050503679636038606, ’47.7%’),

 Text(0.698175755054455, -0.05050361426810683, ’52.3%’)])

**In[8]:**

!pip install tensorflow-hub

**In[9]:**

**!pip install -q tf-models-official**

**In[10]:**

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

Import tensorflow as tf

Import tensorflow\_hub as hub

Import tensorflow\_text as text

From official.nlp import optimization

Tfhub\_handle\_preprocess = ‘https://tfhub.dev/tensorflow/bert\_en\_uncased\_preprocess/3’

Tfhub\_handle\_encoder = ‘https://tfhub.dev/tensorflow/small\_bert/bert\_en\_uncased\_L-4\_H-512\_A-8/1’

**Split data to train, validation, test set.**

**In[11]:**

X\_train, x\_test, y\_train, y\_test = train\_test\_split(dataset[‘title’],dataset[‘target’],test\_size = 0.3, random\_state = 42)

X\_val, xval\_test, y\_val, yval\_test = train\_test\_split(x\_test,y\_test, test\_size =0.5,random\_state = 42)

**This is the classifier model that uses BERT model. This code block is taken from here.**

**In[12]:**

Def build\_classifier\_model():

Text\_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name=’text’)

Preprocessing\_layer = hub.KerasLayer(tfhub\_handle\_preprocess, name=’preprocessing’)

Encoder\_inputs = preprocessing\_layer(text\_input)

Encoder = hub.KerasLayer(tfhub\_handle\_encoder, trainable=True, name=’BERT\_encoder’)

Outputs = encoder(encoder\_inputs)

Net = outputs[‘pooled\_output’]

Net = tf.keras.layers.Dropout(0.1)(net)

Net = tf.keras.layers.Dense(1, activation=None, name=’classifier’)(net)

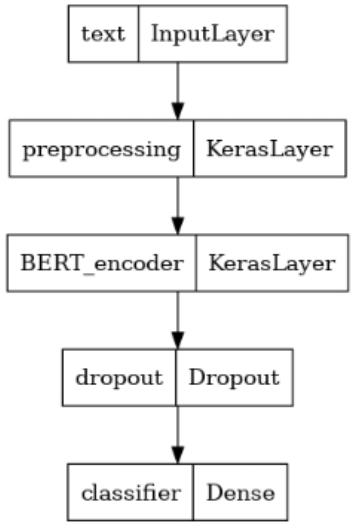
Return tf.keras.Model(text\_input, net)

**In[13]:**

Classifier\_model = build\_classifier\_model()

**In[14]:**

Tf.keras.utils.plot\_model(classifier\_model)

**Out[14]:**

**In[15]:**

Loss = tf.keras.losses.BinaryCrossentropy(from\_logits=True)

Metrics = tf.metrics.BinaryAccuracy()

**In[16]:**

Epochs = 8

Steps\_per\_epoch = 10

Num\_train\_steps = steps\_per\_epoch \* epochs

Num\_warmup\_steps = int(0.1\*num\_train\_steps)

Init\_lr = 1e-5

Optimizer = optimization.create\_optimizer(init\_lr=init\_lr,

Num\_train\_steps=num\_train\_steps,

Num\_warmup\_steps=num\_warmup\_steps,

Optimizer\_type=’adamw’)

**In[17]:**

Classifier\_model.compile(optimizer=optimizer,

Loss=loss,

Metrics=metrics)

**In[18]:**

Print(f’Training model with {tfhub\_handle\_encoder}’)

History = classifier\_model.fit(x=x\_train,y=y\_train,

Validation\_data=(x\_val,y\_val),

Epochs=epochs)

**In[19]:**

Loss, accuracy = classifier\_model.evaluate(x\_test)

Print(f’Loss: {loss}’)

Print(f’Accuracy: {accuracy}’)

**In[20]:**

History\_dict = history.history

Print(history\_dict.keys())

Acc = history\_dict[‘binary\_accuracy’]

Val\_acc = history\_dict[‘val\_binary\_accuracy’]

Loss = history\_dict[‘loss’]

Val\_loss = history\_dict[‘val\_loss’]

Epochs = range(1, len(acc) + 1)

Fig = plt.figure(figsize=(10, 6))

Fig.tight\_layout()

Plt.subplot(2, 1, 1)

# r is for “solid red line”

Plt.plot(epochs, loss, ‘r’, label=’Training loss’)

# b is for “solid blue line”

Plt.plot(epochs, val\_loss, ‘b’, label=’Validation loss’)

Plt.title(‘Training and validation loss’)

Plt.xlabel(‘Epochs’)

Plt.ylabel(‘Loss’)

Plt.legend()

Plt.subplot(2, 1, 2)

Plt.plot(epochs, acc, ‘r’, label=’Training acc’)

Plt.plot(epochs, val\_acc, ‘b’, label=’Validation acc’)

Plt.title(‘Training and validation accuracy’)

Plt.xlabel(‘Epochs’)

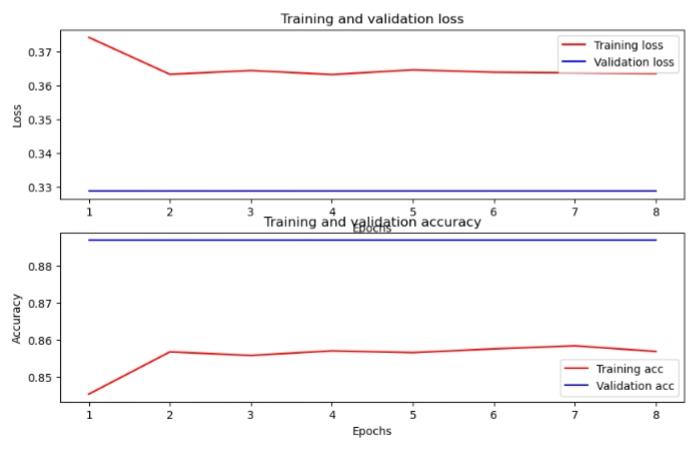
Plt.ylabel(‘Accuracy’)

Plt.legend(loc=’lower right’)

Dict\_keys([‘loss’, ‘binary\_accuracy’, ‘val\_loss’, ‘val\_binary\_accuracy’])

**Out[20]:**

<matplotlib.legend.Legend at 0x7c1f70136b90>



**CONCLUSION:**

In conclusion, fake news detection using the BERT (Bidirectional Encoder Representations from Transformers) method represents a significant advancement in the field of Natural Language Processing (NLP). BERT, with its deep understanding of contextual language, has revolutionized the way we discern deceptive information from factual content.