FAKE NEWS DETECTION USING NNLP (ANALYSIS AND DATASETS)

By- Mathivanan MR -411421205024
(B.TECH/Information Technology,3rd year)
Domain name: Artificial Intelligence

Project: To desing a fake news detection using NLP(Datasets& analysis):

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.

Analysis needed for fake news detection using NLP

- To perform fake news detection using NLP, you can follow these steps:

Data Preprocessing:

Tokenize the text into words or subwords.

Remove stop words, punctuation, and special characters.

Convert text to lowercase for consistency.

Feature Extraction:

Utilize techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings (Word2Vec, GloVe) to represent text.Extract features such as n-grams or character-level features.

Text Vectorization:

Convert the processed text into numerical vectors that can be fed into machine learning models.

Consider using techniques like Bag-of-Words or TF-IDF vectorization.

Model Selection:

regression, random forests, or deep learning models like LSTM or BERT).

Train the model on labeled data (real vs. fake news).

Evaluation Metrics:

Assess the model's performance using metrics like accuracy, precision, recall, F1 score, and area under the ROC curve.

Use a validation set to tune hyperparameters and avoid overfitting.

Cross-Validation:

Implement cross-validation to ensure robust model performance.

- **Feature Importance Analysis:**

If applicable, analyze feature importance to understand which words or features contribute most to the classification.

Ensemble Methods:

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

Handling Imbalanced Data:

Address any class imbalance issues by using techniques such as oversampling, undersampling, or synthetic data generation.

Incorporate External Knowledge:

Leverage external sources like fact-checking databases to enhance the model's ability to identify fake news.

Fine-Tuning and Optimization:

Continuously fine-tune the model based on performance and explore optimization techniques.

Deployment:

Deploy the trained model in a production environment for real-time or batch processing.

Remember to keep the model updated with new data to adapt to evolving patterns of fake news. Regularly evaluate and refine your model's performance over time.

- **Data set for fake news detection using NLP :**

For fake news detection using NLP, you can explore datasets like:

Fake News Dataset (Kaggle): Contains a collection of fake and real news articles. real.

FakeNewsNet : A dataset with multimedia content (text, images, and videos) for fake news detection.

LIAR-PLUS Dataset : Focuses on fact-checking with labeled statements as true, false, or somewhere in between.

Political Social Media Posts : Dataset focused on political fake news on social media.

Remember to review and cite the appropriate sources and adhere to licensing agreements when using these datasets.

- Table of Contents
- Getting the Text Data Ready
- Visualizing the Data
- Cleaning the data
- Frequent Words
- Tokenization
- Building our Model
- Analyzing our Model

Summary

In this kernel , I try to analyse and then build a model to predict whether the news given to us is fake or not

Fake news picture

Here are the things I will try to cover in this Notebook:

Basic EDA of the text data.

Data cleaning

Making some awesome Word cloudsUsing Glove embedding and tokenizer

Building our model

I highly appreciate your feedback, there might be some areas can be fixed or improved.

- **Getting the Data Ready**

Importing necessary libraries

First off , we will import all the necessary libraries we need.our text data.

For our model building i will be using embedding , lstm , dropout and dense layers.I will also be using ReduceLRonPlateau as callback.

- **IMPORTING LIBRARIES:**

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_style('darkgrid')

import nltk
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from wordcloud import STOPWORDS,WordCloud
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
from bs4 import BeautifulSoup
import re,string,unicodedata
```

```
from keras.preprocessing import text,sequence
from nltk.tokenize.toktok import ToktokTokenizer
from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
from string import punctuation
from nltk import pos_tag
from nltk.corpus import wordnet
import keras
from keras.models import Sequential
from keras.layers import LSTM,Dense,Dropout,Embedding
from keras.callbacks import ReduceLROnPlateau
import tensorflow as tf
```

- **Loading the data**

```
real_news=pd.read_csv('../input/fake-and-real-news-dataset/True.csv')
fake_news=pd.read_csv('../input/fake-and-real-news-dataset/Fake.csv')
```

Let's take a sneak peak at our data !

```
real_news.head()
title    text    subject         date
0        As U.S. budget fight looms, Republicans flip t...       WASHINGTON (Reuters) -
The head of a conservat...          politicsNewsDecember 31, 2017
1        U.S. military to accept transgender recruits o...       WASHINGTON (Reuters) -
Transgender people will...          politicsNewsDecember 29, 2017
2        Senior U.S. Republican senator: 'Let Mr. Muell...       WASHINGTON (Reuters) -
The special counsel inv... politicsNewsDecember 31, 2017
3        FBI Russia probe helped by Australian diplomat...    WASHINGTON (Reuters) -
Trump campaign adviser ...        politicsNewsDecember 30, 2017
4        Trump wants Postal Service to charge 'much mor... SEATTLE/WASHINGTON
(Reuters) - President Donal...     politicsNewsDecember 29, 2017
```

```
fake_news.head()
title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year’...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama’s Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017
```

We will now combine both of these data and add a column of 'Isfake' so that we can use all the data as once and the 'Isfake' column will also be our target column

```
real_news['Isfake']=0
fake_news['Isfake']=1
```

Using conactenate function of pandas :

```
df=pd.concat([real_news,fake_news])
```

So how does our data look now ?

```
df.sample(5)
title	text	subject	date	Isfake
19919	COMMUNIST George Soros Says Trump Will Win Pop...	George Soros: Here I have to confess to a lit...	left-news	Sep 26, 2016	1
17566	BREAKING NEWS: Leftist Media Publishes Major F...	How many times in one week can ABC News publis...	left-news	Dec 5, 2017	1
12093	Brexit will not be derailed, says May ahead of...	LONDON (Reuters) - Prime Minister Theresa May ...	worldnews	December 17, 2017	0
15561	Catalonia's ex-leader granted freedom to campa...	BRUSSELS/MADRID (Reuters) - Catalonia s former...	worldnews	November 6, 2017	0
11132	LIBERAL MEDIA IGNORES MELANIA’S Visit To Home ...	First Lady Melania Trump visits HomeSafe, phot...	politics	Apr 15, 2017	1
```

Are there any null values?

```
df.isnull().sum()
title      0
text       0
subject    0
date       0
Isfake     0
dtype: int64
```

As there are no null values , we are saved from the hassle of making up for the missing values. Now we will visualize the data.

- **Visualizing the data**

How many of the given news are fake and how many of them are real?

```
sns.countplot(df.Isfake)
<matplotlib.axes._subplots.AxesSubplot at 0x7f1bce38fc90>
```

The number of fake and real news are almost equal.
Now let us see how many unqiue titles are there. Are any of the titles repeated?

```
df.title.count()
44898
```

How many subjects are there ? We can see that using value_counts()

```
df.subject.value_counts()
politicsNews       11272
worldnews          10145
News                9050
politics            6841
left-news           4459
Government News     1570
US_News              783
Middle-east          778
Name: subject, dtype: int64
```

Let's see how much of the news in different subject are fake !

```
plt.figure(figsize=(10,10))
chart=sns.countplot(x='subject',hue='Isfake',data=df,palette='muted')
chart.set_xticklabels(chart.get_xticklabels(),rotation=90,fontsize=10)
[Text(0, 0, 'politicsNews'),
 Text(0, 0, 'worldnews'),
 Text(0, 0, 'News'),
 Text(0, 0, 'politics'),
 Text(0, 0, 'Government News'),
 Text(0, 0, 'left-news'),
 Text(0, 0, 'US_News'),
 Text(0, 0, 'Middle-east')]
```

Now we will place all of the required columns in one and delete all the not-so-required columns.

```
del df['title']
del df['subject']
del df['date']
```

We are done with this now , we shall head towards cleaning our data!

- **Cleaning the data**

  Our data may consist URLs , HTML tags which might make it difficult for our model to predict properly. To prevent that from happening we will clean our data so as to make our model more efficient.

We will be removing punctuation , stopwords,URLS, html tags from our text data. For this we shall use beautifulsoup and re library which we imported earlier.

```
stop_words=set(stopwords.words('english'))
punctuation=list(string.punctuation)
stop_words.update(punctuation)
def string_html(text):
    soup=BeautifulSoup(text,"html.parser")
    return soup.get_text()

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

def remove_URL(text):
    return re.sub(r'http\S+','',text)

def remove_stopwords(text):
    final_text=[]
    for i in text.split():
        if i.strip().lower() not in stop_words:
            final_text.append(i.strip())
    return " ".join(final_text)

def clean_text_data(text):
    text=string_html(text)
    text=remove_square_brackets(text)
    text=remove_stopwords(text)
    text=remove_URL(text)
    return text
```

Now that we have defined the cleaning functions , let us use em' on our text data.

```
df['text']=df['text'].apply(clean_text_data)
```

We are all done with cleaning and have with us cleaned text data now.Next up are some awesome wordclouds.

- **Frequent Words:**

I wonder what words were the most used in fake news and real news .
So let's see what these frequent words are , and for that we will use wordcloud.

Fake news

```
plt.figure(figsize=(20,20))
wordcloud=WordCloud(stopwords=STOPWORDS,height=600,width=1200).generate(" ".join(df[df.Isfake==1].text))
plt.imshow(wordcloud,interpolation='bilinear')
```

```
<matplotlib.image.AxesImage at 0x7f1bcc7dd810>
```

Real news

```
plt.figure(figsize=(20,20))
wordcloud=WordCloud(stopwords=STOPWORDS,height=600,width=1200).generate(" ".join(df[df.Isfake==0].text))
plt.imshow(wordcloud,interpolation='bilinear')
<matplotlib.image.AxesImage at 0x7f1bcc7cde10>
```

Those were some nice wordclouds , and clearly Donald Trump , United States , etc were very frequent.

- **Tokenization:**

We shall now tokenize our data ,i.e convert the text data into vectors.

```
X_train,X_test,y_train,y_test=train_test_split(df.text,df.Isfake,random_state=0)
max_len=300
```

To tokinize our data , I am using tokenizer here. There are other ways to tokenize data , you can also try them out.

First we initialized the tokenizer with it's size being 10k.
Then we fit the training data on this tokenizer.
Then we convert the text to sequences and save it in X_train variable.
Lastly we add a padding layer around our sequence.
Here is a example of what tokenizer does

```
tokenizer=text.Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(X_train)
tokenizer_train=tokenizer.texts_to_sequences(X_train)
X_train=sequence.pad_sequences(tokenizer_train,maxlen=max_len)
tokenizer_test=tokenizer.texts_to_sequences(X_test)
X_test=sequence.pad_sequences(tokenizer_test,maxlen=max_len)
```

Now we will import our GLOVE file , I am using the 100d version here.

```
glove_file='../input/glove-twitter/glove.twitter.27B.100d.txt'
```

Now we will get the coefficients from the glove file and save it in embedding index variable.

```
def get_coefs(word, *arr):
   return word, np.asarray(arr,dtype='float32')
embeddings_index=dict(get_coefs(*o.rstrip().rsplit(' ')) for o in
open(glove_file,encoding="utf8"))
```

What's happening in the next code tab:
We first take all the values of the embeddings and store it in all_embs.
Then we take the mean and standard deviation of all the embeddings.
We now take the word indices using .word_index function of tokenizer.
Then we will see what the length of each vector would be and save it in nb_words.
We make an embedding matrix.

```
all_embs=np.stack(embeddings_index.values())
emb_mean,emb_std=all_embs.mean(),all_embs.std()

word_index=tokenizer.word_index
nb_words=min(max_features,len(word_index))

embedding_matrix =
np.random.normal(emb_mean,emb_std,(nb_words,emb_size))

for word,i in word_index.items():
   if i>=max_features: continue
   embedding_vector=embeddings_index.get(word)
   if embedding_vector is not None: embedding_matrix[i]=embedding_vector
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3254:
FutureWarning: arrays to stack must be passed as a "sequence" type such as list or
tuple. Support for non-sequence iterables such as generators is deprecated as of
NumPy 1.16 and will raise an error in the future.
  if (await self.run_code(code, result,  async_=asy)):
```

- **Building our model**

We have succesfully done the tokenization part , let's build our model now!

Here are the parameters I'm taking.

```
batch_size=256
epochs=10
emb_size=100
```

Let's initialize our callback.

```
leaning_rate_reduction=ReduceLROnPlateau(monitor='val_accuracy',patience=2,verbose=10,factor=0.5,min_lr=0.00001)
```

Let's build our model.Here are the layers I'm using:

Starting with an embedding layer
Then 3 LSTM layers
Then 2 Dense layers
I am using Adam optimizer for our model.

```
model=Sequential()
model.add(Embedding(max_features,output_dim=emb_size,weights=[embedding_matrix],input_length=max_len,trainable=False))
model.add(LSTM(units=256,return_sequences=True,recurrent_dropout=0.25,dropout=0.25))
model.add(LSTM(units=128,return_sequences=True,recurrent_dropout=0.25,dropout=0.25))
model.add(Dense(units=32,activation='relu'))
model.add(Dense(1,'sigmoid'))
model.compile(optimizer=keras.optimizers.Adam(lr=0.01),loss='binary_crossentropy',metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________
Layer (type)              Output Shape           Param #
=================================================================
embedding (Embedding)       (None, 300, 100)        1000000

_________________________
lstm (LSTM)               (None, 300, 256)        365568
```

```
________________________
lstm_1 (LSTM)           (None, 300, 128)        197120
________________________
lstm_2 (LSTM)           (None, 64)            49408
________________________
dense (Dense)           (None, 32)            2080
________________________
dense_1 (Dense)          (None, 1)             33
=================================================================
Total params: 1,614,209
Trainable params: 614,209
Non-trainable params: 1,000,000
```

- **Training our Model:**

```
history=model.fit(X_train,y_train,batch_size=batch_size,validation_data=(X_test,y_test),epochs=epochs,callbacks=[leaning_rate_reduction])
Epoch 1/10
accuracy: 0.8259 - val_loss: 0.0776 - val_accuracy: 0.9769 - lr: 0.0100
Epoch 2/10
132/132 [==============================] - 451s 3s/step - loss: 0.0360 -
accuracy: 0.9894 - val_loss: 0.0184 - val_accuracy: 0.9933 - lr: 0.0100
Epoch 3/10
132/132 [==============================] - 458s 3s/step - loss: 0.0237 -
accuracy: 0.9929 - val_loss: 0.0155 - val_accuracy: 0.9952 - lr: 0.0100
Epoch 4/10
132/132 [==============================] - 457s 3s/step - loss: 0.0127 -
accuracy: 0.9955 - val_loss: 0.0174 - val_accuracy: 0.9941 - lr: 0.0100
Epoch 5/10
132/132 [==============================] - 453s 3s/step - loss: 0.0138 -
accuracy: 0.9958 - val_loss: 0.0144 - val_accuracy: 0.9962 - lr: 0.0100
Epoch 6/10
132/132 [==============================] - 460s 3s/step - loss: 0.0079 -
accuracy: 0.9975 - val_loss: 0.0117 - val_accuracy: 0.9972 - lr: 0.0100
```

```
Epoch 7/10
132/132 [==============================] - 461s 3s/step - loss: 0.0081 -
accuracy: 0.9976 - val_loss: 0.0147 - val_accuracy: 0.9955 - lr: 0.0100
Epoch 8/10
132/132 [==============================] - ETA: 0s - loss: 0.0051 - accuracy:
0.9983
Epoch 00008: ReduceLROnPlateau reducing learning rate to
0.004999999888241291.
132/132 [==============================] - 461s 3s/step - loss: 0.0051 -
accuracy: 0.9983 - val_loss: 0.0103 - val_accuracy: 0.9972 - lr: 0.0100
Epoch 9/10
132/132 [==============================] - 454s 3s/step - loss: 0.0035 -
accuracy: 0.9989 - val_loss: 0.0088 - val_accuracy: 0.9979 - lr: 0.0050
Epoch 10/10
132/132 [==============================] - 455s 3s/step - loss: 0.0035 -
accuracy: 0.9989 - val_loss: 0.0075 - val_accuracy: 0.9985 - lr: 0.0050
```

Let's see our model in action ! ;)

```
pred = model.predict_classes(X_test)
pred[5:10]
array([[0],
      [1],
      [0],
      [1]], dtype=int32)
```

- **Analyzing our model**

Let's see how the accuracy and loss graphs of our model look now !

```
epochs = [i for i in range(10)]
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
fig.set_size_inches(20,10)
```

```
ax[0].plot(epochs,train_acc,'go-',label='Training Accuracy')
ax[0].plot(epochs,val_acc,'ro-',label='Validation Accuracy')
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Accuracy')
ax[0].legend()

ax[1].plot(epochs,train_loss,'go-',label='Training Loss')
ax[1].plot(epochs,val_loss,'ro-',label='Validation Loss')
ax[1].set_xlabel('Loss')
ax[1].set_ylabel('Accuracy')
ax[1].legend()

plt.show()
```

We will now see how many of the samples were wrongly predicted using the confusion matrix.

```
cm
Fake   Not Fake
Fake   5353  14
Not Fake      3        5855
plt.figure(figsize=(10,10))
sns.heatmap(cm,cmap="Blues",linecolor='black',linewidth=1,annot=True,fmt='',xticklabels=['Fake','Not Fake'],yticklabels=['Fake','Not Fake'])
plt.xlabel('Actual')
plt.ylabel('Predicted')
Text(69.0, 0.5, 'Predicted')
```

Now what is our accuracy on Test and Train set?

```
print(f'Accuracy of the model on Training Data is - {
model.evaluate(X_train,y_train)[1]*100:.2f}')
print(f'Accuracy of the model on Testing Data is -
{model.evaluate(X_test,y_test)[1]*100:.2f}')
```

```
1053/1053 [==============================] - 219s 208ms/step - loss:
4.7703e-04 - accuracy: 0.9999
Accuracy of the model on Training Data is - 99.99
351/351 [==============================] - 72s 206ms/step - loss: 0.0075 -
accuracy: 0.9985
Accuracy of the model on Testing Data is -  99.85
```

Conclusion:

In conclusion these are the following datasets for predicting fake news detection And by following the mentioned analysis we can analysis the fake news inorder to predict them.