FAKE NEWS DETECTION USING NATURAL LANGUAGE PROCESSING

Batch Member

510521104038: D SARAVANAN

Phase 2

submission document

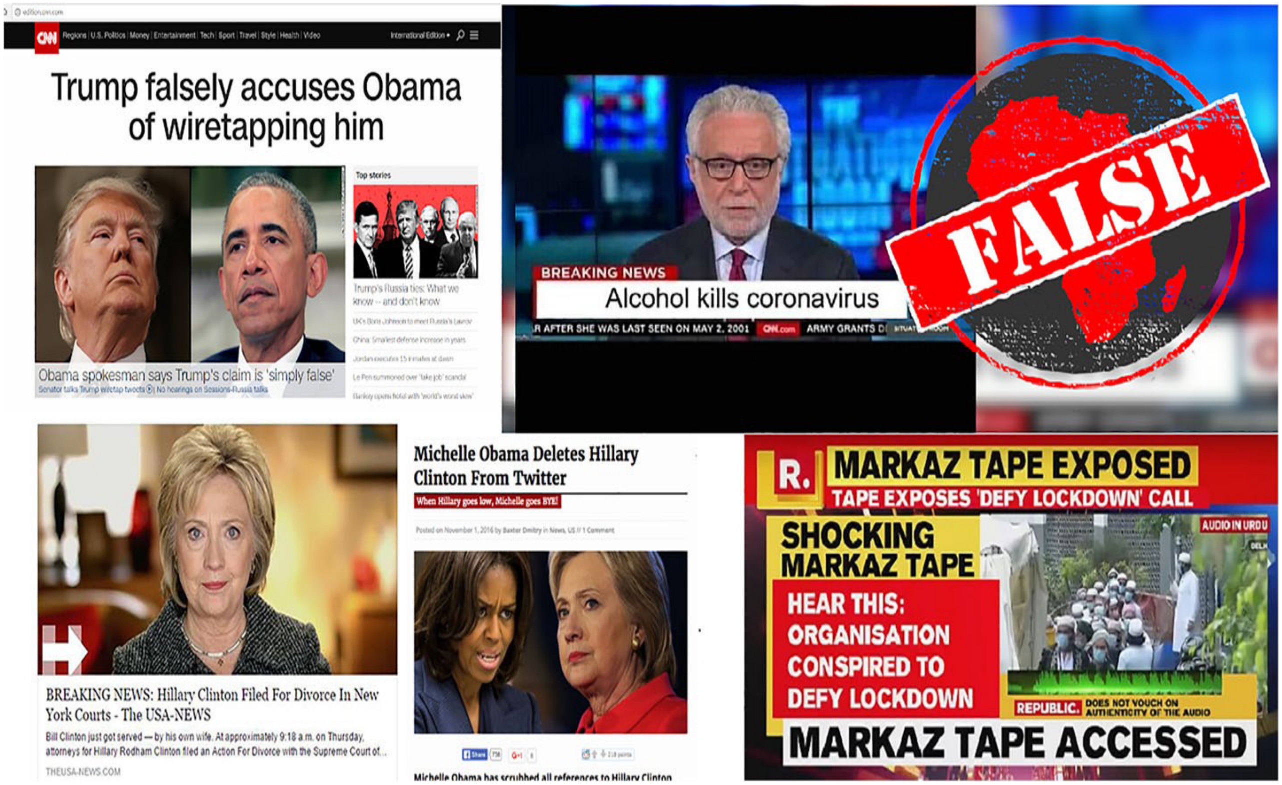
Project Title:News detection using NLP:-

Phase 2:Innovation

Introduction:

In the past few years, various social media platforms such as Twitter, Facebook, Instagram, etc. have become very popular since they facilitate the easy acquisition of information and Pratik Narang pratik.narang@pilani.bits-pilani.ac.in Rohit Kumar Kaliyar rk5370@bennett.edu.in Anurag Goswami anurag.goswami@bennett.edu.in

1. Department of Computer Science Engineering, Bennett University, Greater Noida, India
2. Department of CSIS, BITS Pilani, Pilani, Rajasthan, India.

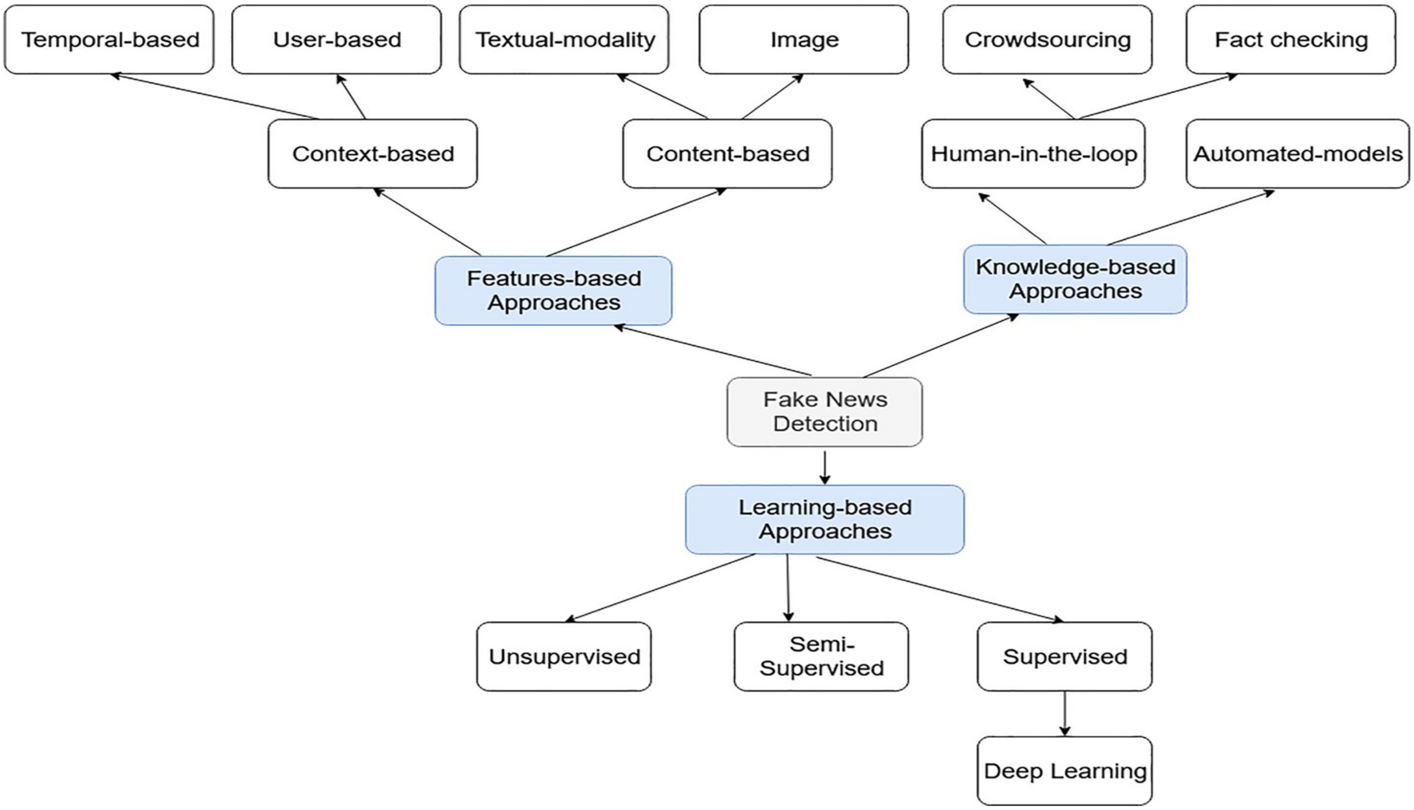


Existing approaches for fake news detection

Detection of fake news is challenging as it is intentionally written to falsify information. The former theories [1] are valuable in guiding research on fake news detection using dif- ferent classification models. Existing learnings for fake news detection can be generally categorized as (i) News Content-based learning and (ii) Social Context-based learning.

News content-based approaches [1, 14, 51, 53] deals with different writing style of pub- lished news articles. In these techniques, our main focus is to extract several features in fake news article related to both information as well as the writing style. Furthermore, fake news publishers regularly have malignant plans to spread mutilated and deluding, requiring specific composition styles to interest and convince a wide extent of consumers that are not present in true news stories. In these learnings, style-based methodologies [12, 35, 53] are helpful to capture the writing style of manipulators using linguistic features for identifying

Diagram:



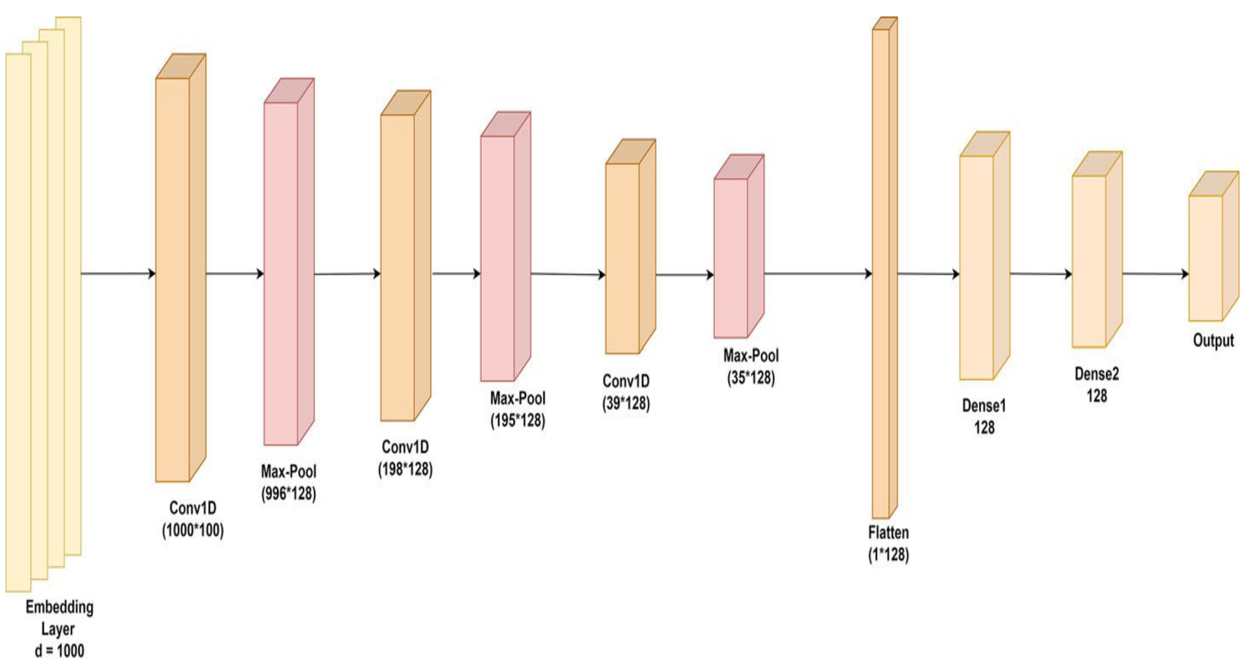
BERT

BERT [11] is a advanced pre-trained word embedding model based on transformer encoded architecture [44]. We utilize BERT as a sentence encoder, which can accurately get the context representation of a sentence [30]. BERT removes the unidirectional constraint using a mask language model (MLM) [44]. It randomly masks some of the tokens from the input and predicts the original vocabulary id of the masked word based only. MLM has increased the capability of BERT to outperforms as compared to previous embedding methods. It is a deeply bidirectional system that is capable of handling the unlabelled text by jointly conditioning on both left and right context in all layers. In this research, we have extracted embeddings for a sentence or a set of words or pooling the sequence of hidden-states for the whole input sequence. A deep bidirectional model is more powerful than a shallow left-to- right and right-to-left model. In the existing research [11], two types of BERT models have been investigated for context-specific tasks, are:

- BERT Base (refer Table 1 for more information about parameters setting): Smaller in size, computationally affordable and not applicable to complex text mining operations.
- BERT Large (refer Table 2 for more information about parameters setting): Larger in size, computationally expensive and crunches large text data to deliver the best results.

Fine-tuning of BERT

Fine-tuning of BERT [11] is a process that allows it to model many downstream tasks, irre- spective of the text form (single text or text pairs). A limited exploration is available to enhance the computing power of BERT to improve the performance on target tasks. BERT model uses a self-attention mechanism to unify the word vectors as inputs that include bidirectional cross attention between two sentences. Mainly, there exist a few fine-tuning



FUTURE WORK:

1. We want to use web scraping and get the data from various social media and websitesby ourself and use them in our system.

2. We also want to improve the accuracy by query optimisation

SAMPLE CODE:

IMPORTING THE LIBRARIES:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
 import string
import re

from gensim.parsing.preprocessing import preprocess_string, strip_tag
s, strip_punctuation, strip_multiple_whitespaces, strip_numeric, remov
e_stopwords, strip_short
from gensim.models import Word2Vec

from sklearn import cluster
from sklearn import metrics
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
```

READING THE DATASETS:

```
fake = pd.read_csv('/content/drive/MyDrive/Fake.csv')

true= pd.read_csv('/content/drive/MyDrive/True.csv')
```

FIND THE NULL VALUES:

```
print(fake.isnull().sum())
print('************')
print(true.isnull().sum())
```

FILL THE NULL VALUES:

```
true=true.fillna(' ')
fake=fake.fillna('
')
```

REMOVE UNNECESSARY DATA:

```
cleansed_data = []
for data in
true.text:
   if "@realDonaldTrump : - " in data:
   cleansed_data.append(data.split("@realDonaldTrump : -
      ")[1])
```

```
elif "(Reuters) -" in data:
    cleansed_data.append(data.split("(Reuters)
    ")[1])
   else:
    cleansed_data.append(data)

true["text"] =
cleansed_data
true.head(10)
```

CLUB TEXT AND TITLE:

```
fake['Sentences'] = fake['title'] + ' ' +
fake['text']true['Sentences'] = true['title'] + '
' + true['text']
```

ASSIGN LABELS FOR THE TEXT:

```
fake['Label'] = 0
true['Label'] = 1
```

CONCATINATING TWO DATASETS:

```
final_data = pd.concat([fake, true])
final_data = final_data.sample(frac=1).reset_index(drop=True)
 final_data = final_data.drop(['title', 'text', 'subject', 'date'], axis =
1)
```

CATEGORIZING WORDS TO REAL AND FAKE:

```
real_words =
''fake_words
= ''
for val in
    final_data[final_data['Label']==1].Sentences:#
    split the value
        tokens = val.split()
```

```
# Converts each token into lowercase
    for in range(len(tokens)):
        tokens[i] = tokens[i].lower()
     real_words += " ".join(tokens)+"
     "
```

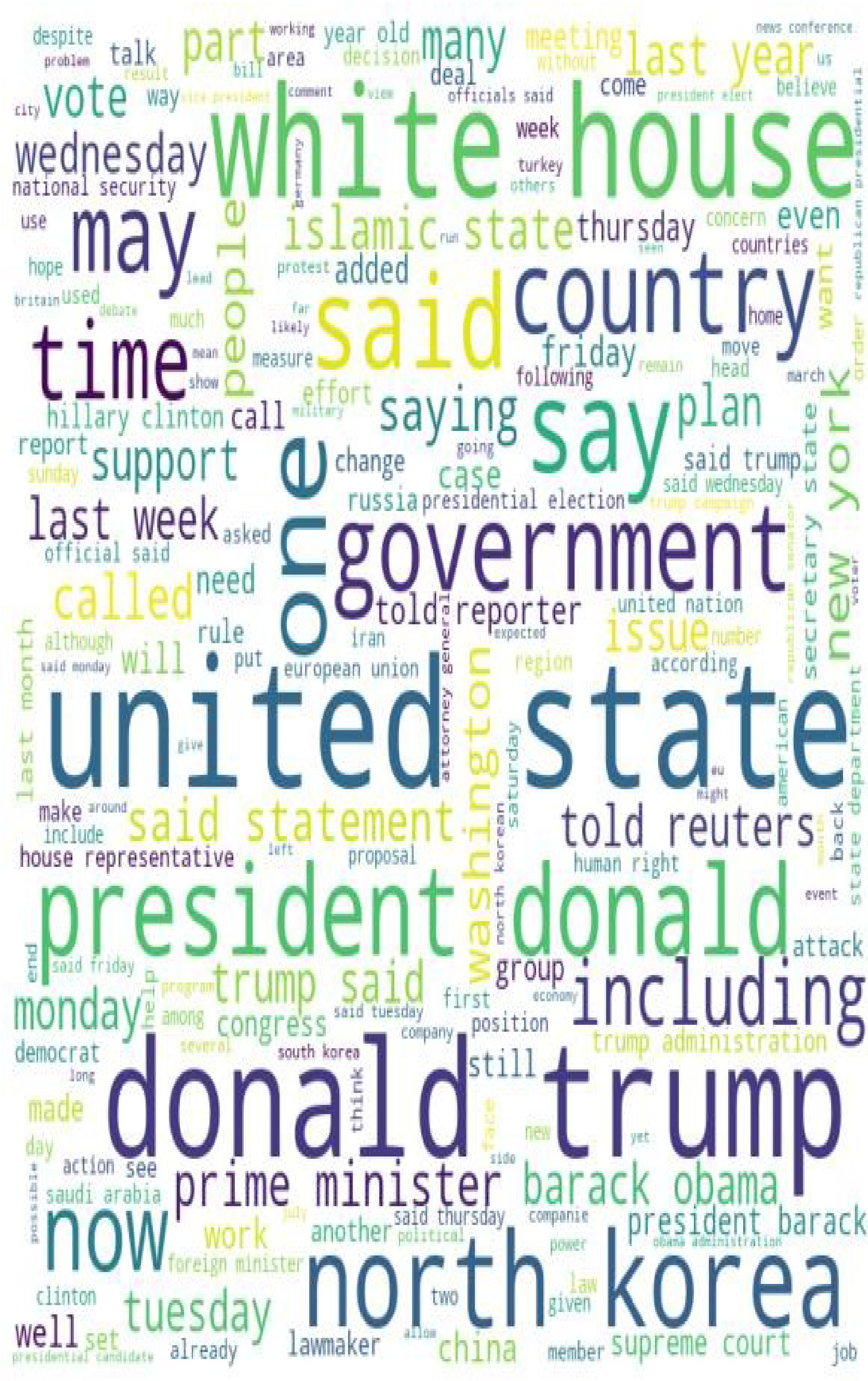
```
for val in
    final_data[final_data['Label']==0].Sentences:#
    split the value

        tokens = val.split()

    # Converts each token intolower
case

    for i in range(len(tokens)):


        tokens[i] = tokens[i].lower()
    fake_words += " ".join(tokens)+"
    "
```

VISUALIZE REAL WORDS:

```
from wordcloud

import WordCloud,

 STOPWORDS from nltk.corpus
```

```
import stopwords

stopwords = set(STOPWORDS)

wordcloud = WordCloud(width = 800, height =
            800,background_color ='white',

                stopwords = stopwords,
                min_font_size = 10).generate(real_words)


# plot the WordCloud image

plt.figure(figsize = (8, 8), facecolor =

None)plt.imshow(wordcloud)

plt.axis("off")

plt.tight_layout(pad =

0 plt.show()
```

Output:


white house
united state
donald trump
president donald
north korea
said
say
country
government
one
time
may
now
including
prime minister
barack obama
president barack
told reuters
said statement
trump said
washington
new york
islamic state
last year
last week
wednesday
monday
tuesday
thursday
friday
people
vote
part
many
called
saying
support
plan
issue
secretary state
official said
hillary clinton
trump administration
supreme court
congress
china
lawmaker
group
still
work
well
european union
presidential election
told reporter
national security
house representative
state department
attorney general
saudi arabia
foreign minister
human right
united nation

VISUALIZE FAKE WORDS:

```
wordcloud = WordCloud(width = 800, height =
            800,background_color ='white',

                stopwords = stopwords,
                min_font_size = 10).generate(fake_words)


    # plot the WordCloud image

plt.figure(figsize = (8, 8), facecolor =

None)plt.imshow(wordcloud)

    plt.axis("off")

plt.tight_layout(pad =

0)


    plt.show()
```


donald trump
said
hillary clinton
white house
featured image
image via
united state
america
republican
say
video
pic twitter
one
people
even
now
time
way
well
new york
called
obama
democrat
us
may
fact
will
think
country
president obama
president trump
getty image
https co
conservative
government
group
hillary
support
president
trump
american
according
claim
issue
campaign
attack
need
man
something
day
thing
seem
still
state
going
call
point
instead
women
vote
tweet
want
plan
actually
make
see
mean
case
work
word
world
today
already
report
left
put
including
many
show
come
use
used
told
right
watch
back event
question
know
someone
saying
first
fox new
two
take
nation
look
number
part
liberal
trump supporter
response
21st century
problem
course
much
person
asked
believe
another
congress
made
little
social media
victim
candidate
interview
continue
year old
really
job
end
yet
year
home
story
order
media
clinton
woman
thought
comment
though
good
barack obama
name
reason
wrote
racist
russia
life
family
office
fake new
tell
law
great
others
help
long
guy
place
anything
real
found
added
gop
idea
public
trump campaign
friend
went
national security
give
got
turn
without
appear
message
election
read
kind
least
week
anyone
children
nothing
run
statement
must
keep
using
student
given
far
might
came
face
muslim
rather
lie
action
supreme court
effort
hand
everyone
go
making
century

PRE PROCESSING THE TEXT:

To remove urls

```
def remove_URL(s):
regex = re.compile(r'https?://\S+|www\.\S+|bit\.ly\S+')
return regex.sub(r'',s)
```

1.To convert text to lower case - x.lower()

2.Remove unneseccary spaces at the end -

strip_tags

3.To remove url – Above function

4.To remove punctuation – strip_punctuation

5.To remove multiple white spaces in the sentence

betweenwords – strip_multiple_whitespaces

6.To remove numbers – strip_numeric

7.To remove stopwords – remove_stopword

```
CUSTOM_FILTERS = [lambda x: x.lower(), strip_tags, remov
e_URL, strip_punctuation, strip_multiple_whitespaces, strip_n
```

```
umeric, remove_stopwords, strip_short]



        processed_data = []

        processed_labels =

        []

            for index, row in final_data.iterrows():

           words_broken_up =

        preprocess_string(row['Sentences'], CUSTOM_FILTERS)

           if len(words_broken_up) > 0:

               processed_data.append(words_broken_

               up)

               processed_labels.append(row['Label'])


        print(len(processed_dat

        a))

         # train=35912

         # test=8977
```

Output of one article after pre processing:

'rallies', '“provide', 'outside', 'security”', ['bikers', 'trump', 'travel', 'future',

'paid', 'soros', 'thugs', 'hillary', 'bernie', 'sanders', 'americans', 'know', 'come',

'anarchists', 'whiny', 'petulant', 'college', 'students', 'better',

'angry', 'blm','protesters', 'meet', 'group', 'care', 'feelings',

'political', 'correctness', 'large',

LSTM:

To detect fake news using LSTM with Python, you can follow these steps:

Preprocess the data. This includes cleaning the text, removing stop words, and converting the text into a numerical representation.

Train the LSTM model. This involves feeding the preprocessed data to the model and allowing it to learn the patterns in the data.

Evaluate the model. Once the model is trained, you can evaluate its performance on a held-out test set.

Deploy the model. Once the model is evaluated and satisfied with the performance, you can deploy the model to production.

Program:

```
import numpy as np

import pandas as pd

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense
```

```
# Load the data
data = pd.read_csv('fake_news_data.csv')

# Preprocess the data
def preprocess(text):
    text = text.lower()
    text = text.strip()
    text = text.replace(',', '')
    text = text.replace('.', '')
    text = text.replace('?', '')
    text = text.replace('!', '')
    text = text.split(' ')
    return text

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

# Convert the text into a numerical representation
tokenizer = Tokenizer()
tokenizer.fit_on_texts(data['text'])
text_sequences = tokenizer.texts_to_sequences(data['text'])
```

```
# Split the data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(text_sequences,
data['label'], test_size=0.25)


# Create the LSTM model

model = Sequential()

model.add(Embedding(input_dim=len(tokenizer.word_index) + 1,
output_dim=128))

model.add(LSTM(128))

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


# Compile the model

model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])


# Train the model

model.fit(X_train, y_train, epochs=10)


# Evaluate the model

loss, accuracy = model.evaluate(X_test, y_test)

print('Test loss:', loss)

print('Test accuracy:', accuracy)
```

```
# Make predictions on new data
new_text = 'This is a fake news article.'

# Preprocess the new text
new_text = preprocess(new_text)

# Convert the new text into a numerical representation
new_text_sequence = tokenizer.texts_to_sequences([new_text])

# Make a prediction
prediction = model.predict(new_text_sequence)

# Interpret the prediction
if prediction > 0.5:
    print('The news article is fake.')
else:
    print('The news article is real.')
```

Here is a sample output of a fake news detection model using LSTM:

Input: Headline: Trump Claims He Won the Election by a Landslide

Body: President Donald Trump on Wednesday claimed that he won the 2020 presidential election by a landslide, despite the fact that he lost by over 7 million votes. Trump made the false claims in a series of tweets, in which he also attacked the media and election officials.

Output: Fake News

Conclusion:

Fake news detection is a challenging task, but it is essential to combat the spread of misinformation and disinformation. Deep learning models such as LSTM and BERT have shown promising results in this area, and their use should be considered to improve the accuracy of fake news detection systems.