AI_PHASE 4 IBM

TITLE: FAKE NEWS DETECTION USING

NLP techniques and training a classification model. Text Preprocessing and Feature Extraction, model training and evaluation.

TEAM MEMBERS:	REG NO
1. M. BHARANI DHARAN [LEADER]	731221104302
2. S. ARULMURUGAN.	731221104004
3. S. VASANTH KUMAR.	731221104038
4. S. SURYA.	731221104036
5. V. SRINIVAS.	731221104307

FAKE NEWS DETECTION USING NLP

Continue building the fake news detection model by applying NLP techniques and training a classification model. Text Preprocessing and Feature Extraction, model training and evaluation does the following.

1. LSTM – LONG SHORT TERM MEMORY:

- First load in the data. The preprocessing only consist of normalization and the creation of windows.
- Creation of the LSTM model
- Training the LSTM model
- Testing the LSTM model with 1 time step and with window

```
In [1]: import keras
        from keras.preprocessing import text,sequence
        from keras.models import Sequential
        from keras.layers import Dense, Embedding, LSTM, Dropout
```

2. TEXT PREPROCESSING:

Text preprocessing is essential for cleaning and transforming raw text data into a format suitable for machine learning models.

a) Tokenization.

b) Lowercasing.

c) Removing Special Characters and numbers.

d) Lemmatization or Stemming

A) Tokenization.

```
[ ] import nltk
    from nltk.tokenize import word_tokenize
```

```
[ ] nltk.download('punkt')
    text = "Sample text for tokenization."
    tokens = word_tokenize(text)
    print(tokens)
    lowercase_tokens = [token.lower() for token in tokens]
    print(lowercase_tokens)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
['Sample', 'text', 'for', 'tokenization', '.']
['sample', 'text', 'for', 'tokenization', '.']
```

B) Lowercasing.

```
[ ] lowercase_tokens = [token.lower() for token in tokens]
    print(lowercase_tokens)
```

```
['sample', 'text', 'for', 'tokenization', '.']
```

C) Removing Special Characters and numbers.

```
In [22]: import re
         import string
         first_text = re.sub('\[[^]]*\]', ' ', first_text)
         first_text = re.sub('[^a-zA-Z]',' ',first_text)
         first_text = first_text.lower()
         first_text
```

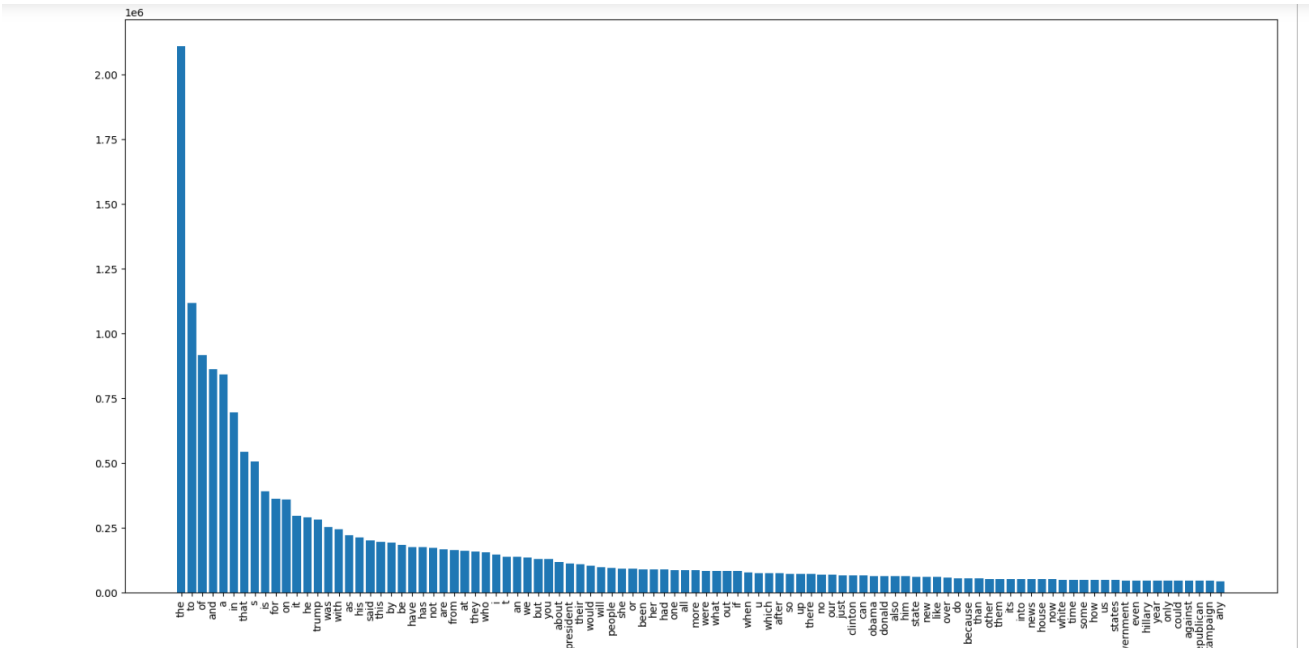
```
Out[22]: 'politicsnews jones certified u s  senate winner despite moore challenge  reuters    alabama officials on thursday certified de
mocrat doug jones the winner of the state s u s  senate race  after a state judge denied a challenge by republican roy moore  w
hose campaign was derailed by accusations of sexual misconduct with teenage girls  jones won the vacant seat by about        vo
tes  or     percentage points  election officials said  that made him the first democrat in a quarter of a century to win a sen
ate seat in alabama   the seat was previously held by republican jeff sessions  who was tapped by u s  president donald trump a
s attorney general  a state canvassing board composed of alabama secretary of state john merrill  governor kay ivey and attorne
y general steve marshall certified the election results  seating jones will narrow the republican majority in the senate to
of     seats  in a statement  jones called his victory  a new chapter  and pledged to work with both parties  moore declined to
concede defeat even after trump urged him to do so  he stood by claims of a fraudulent election in a statement released after t
he certification and said he had no regrets  media outlets reported  an alabama judge denied moore s request to block certifica
tion of the results of the dec     election in a decision shortly before the canvassing board met  moore s challenge alleged th
ere had been potential voter fraud that denied him a chance of victory  his filing on wednesday in the montgomery circuit court
sought to halt the meeting scheduled to ratify jones  win on thursday  moore could ask for a recount  in addition to possible o
ther court challenges  merrill said in an interview with fox news channel  he would have to complete paperwork  within a timed
period  and show he has the money for a challenge  merrill said  we ve not been notified yet of their intention to do that   m
errill said  regarding the claim of voter fraud  merrill told cnn that more than     cases had been reported  we ve adjudicate
d more than    of those  we will continue to do that  he said  republican lawmakers in washington had distanced themselves fr
om moore and called for him to drop out of the race after several women accused him of sexual assault or misconduct dating back
to when they were teenagers and he was in his early  s  moore has denied wrongdoing and reuters has not been able to independ
ently verify the allegations  '
```

D) Lemmatization or Stemming

```
[ ] from nltk.stem import PorterStemmer
    stemmer = PorterStemmer()
    stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
    print(stemmed_tokens)

    ['sampl', 'text', 'token']
```

```
In [8]: word_counts = tokenizer.word_counts
        sorted_word_counts = sorted(word_counts.items(), key=lambda x: x[1], reverse=True)
        most_common_words = sorted_word_counts[:100]
        words, counts = zip(*most_common_words)
        plt.figure(figsize=(20,10))
        plt.bar(words, counts)
        plt.xticks(rotation='vertical')
        plt.show()
```



3. Feature Extraction:

After preprocessing, you need to convert text into numerical features that machine learning models can understand.

Bag of Words (BoW) Representation:

Convert text into a matrix of token counts.

```
[ ]  from sklearn.feature_extraction.text import TfidfVectorizer

     tfidf_vectorizer = TfidfVectorizer()
     X_tfidf = tfidf_vectorizer.fit_transform([" ".join(stemmed_tokens)])
     print(X_tfidf.toarray())
```

```
[[0.57735027 0.57735027 0.57735027]]
```

4.Model Training and Evaluation:

Choose a suitable classification model, train it on the features (X) and corresponding labels (y), and evaluate its performance.

Training the LSTM:

In [12]:
```
features = news_df['text']
targets = news_df['class']

X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size=0.20, random_state=18)
```

In [13]:
```
max_vocab = 10000
tokenizer = Tokenizer(num_words=max_vocab)
tokenizer.fit_on_texts(X_train)

# tokenize the text into vectors i.e. list
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)

X_train = tf.keras.preprocessing.sequence.pad_sequences(X_train, padding='post', maxlen=256)
X_test = tf.keras.preprocessing.sequence.pad_sequences(X_test, padding='post', maxlen=256)
```

In [14]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(max_vocab, 128),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,  return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1)
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 128)         1280000

 bidirectional (Bidirection  (None, None, 128)         98816
 al)

 bidirectional_1 (Bidirecti  (None, 32)                18560
 onal)

 dense (Dense)               (None, 64)                2112

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 1399553 (5.34 MB)
Trainable params: 1399553 (5.34 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [15]:
```
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=2, restore_best_weights=True)
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=10,validation_split=0.1, batch_size=30, shuffle=True, callbacks=[early_stop])
```

```
Epoch 1/10
1078/1078 [==============================] - 430s 388ms/step - loss: 0.2305 - accuracy: 0.8818 - val_loss: 0.0480 - val_accurac
y: 0.9850
Epoch 2/10
1078/1078 [==============================] - 402s 372ms/step - loss: 0.0442 - accuracy: 0.9886 - val_loss: 0.0313 - val_accurac
y: 0.9908
Epoch 3/10
1078/1078 [==============================] - 1211s 1s/step - loss: 0.0207 - accuracy: 0.9954 - val_loss: 0.0258 - val_accuracy:
0.9911
Epoch 4/10
1078/1078 [==============================] - 405s 376ms/step - loss: 0.0094 - accuracy: 0.9983 - val_loss: 0.0194 - val_accurac
y: 0.9930
Epoch 5/10
1078/1078 [==============================] - 409s 379ms/step - loss: 0.0085 - accuracy: 0.9981 - val_loss: 0.0320 - val_accurac
y: 0.9911
Epoch 6/10
1078/1078 [==============================] - 402s 373ms/step - loss: 0.0082 - accuracy: 0.9978 - val_loss: 0.0386 - val_accurac
y: 0.9914
```

In [99]:
```
loss, accuracy,recall= model.evaluate(X_tst, y_tst, verbose=0)

# Print metrics
print('Accuracy  : {:.4f}'.format(accuracy))
print('Recall  : {:.4f}'.format(recall))
```

```
Accuracy  : 0.9915
Recall  : 0.9787
```

In [ ]:
```

```

THANK YOU