SENTIMENT ANALYSIS USING BI-LSTM

-NAVIN R C

-VAIBHAV RAJ DIKSHIT

INTRODUCTION

- **SENTIMENT ANALYSIS:** Sentiment analysis is the process of computationally identifying and categorizing options expressed in a piece of text , especially in order to determine that particular topic is positive , negative or neutral.
- **BI-LSTM NETWORKS:** Bidirectional long short-term memory are a type of Recurrant Neural Network that is capable of capturing long-term dependencies in sequential data bidirectionally.Bi LSTM network are effective fpr analyzing sequence of data making them suitable for sentiment analysis tasks

DATASETS

Here we have taken train test and val datasets which contains sentences along with the emotions such **as love , joy , fear , anger , sadness , etc.**

Now load the data set and covert the sentence into word vectors by using **TOKENIZER** method.

Converting the sentence into word vectors

```
tokenizer=Tokenizer(15212,lower=True,oov_token='UNK')
tokenizer.fit_on_texts(X)
```

```
len(tokenizer.word_index)
```

```
15213
```

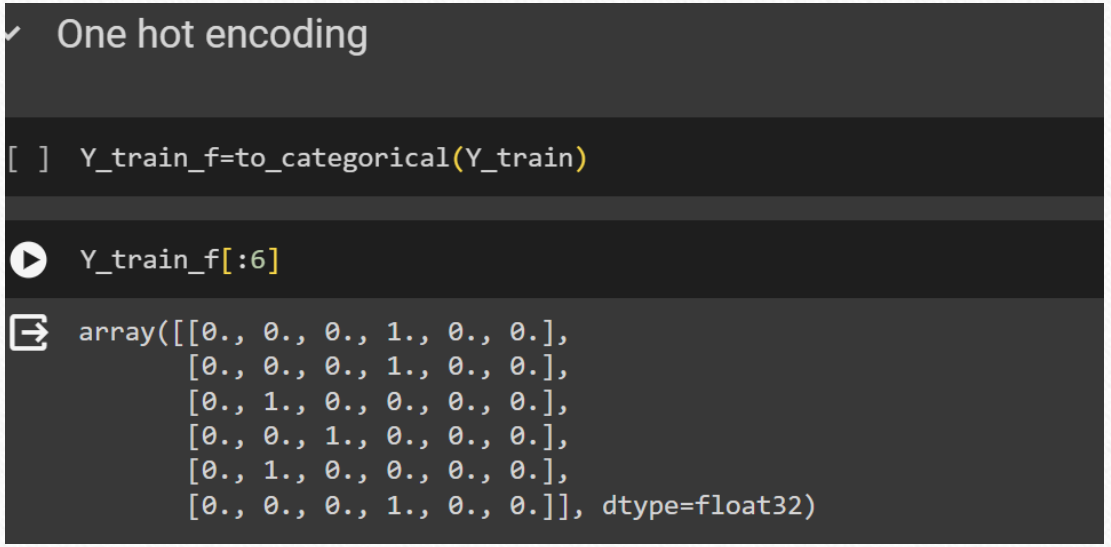
```
X_train=tokenizer.texts_to_sequences(X)
X_train_pad=pad_sequences(X_train,maxlen=80,padding='post')
```

```
df_train['Sentiment']=df_train.Sentiment.replace({'joy':0,'anger':1,'love':2,'sadness':3,'fear':4,'surprise':5})
```

```
Y_train=df_train['Sentiment'].values
```

One-Hot Encoding:

One-hot encoding is a technique used to convert categorical variables (such as words) into binary vectors where each word is represented by a vector of zeros with a single 1 at the index corresponding to the word's position in the vocabulary.



Creating a Bi-LSTM Mode

Creating a Model

```
[ ]  from keras.models import Sequential
     from keras.layers import LSTM,Bidirectional,Dense,Embedding,Dropout
```

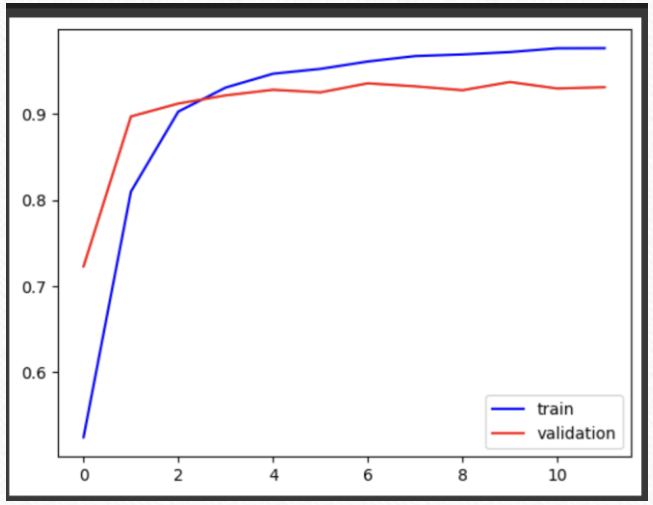
```
model=Sequential()
model.add(Embedding(15212,64,input_length=80))
model.add(Dropout(0.6))
model.add(Bidirectional(LSTM(80,return_sequences=True)))
model.add(Bidirectional(LSTM(160)))
model.add(Dense(6,activation='softmax'))
print(model.summary())
```

```
Model: "sequential_1"
```

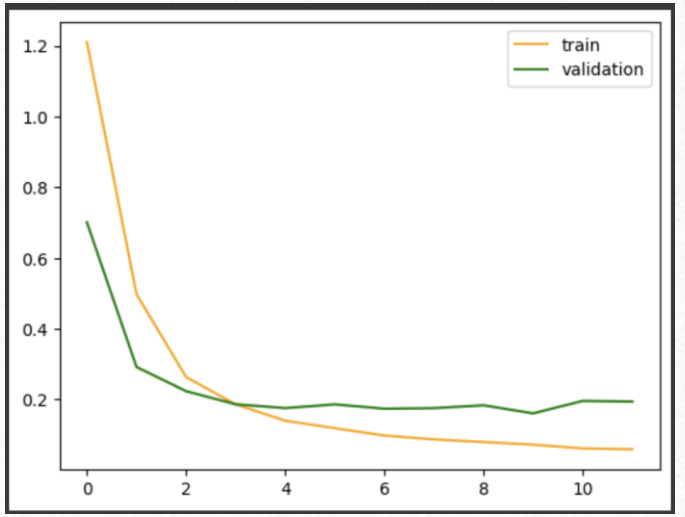
Bi-LSTM networks consist of two LSTM layers, one processing the input sequence in the forward direction and the other processing it in the backward direction.

By processing the input sequence in both directions, Bi-LSTM networks capture contextual information from both past and future words.

Plotting Loss and Accuracy Curves



Accuracy Curves



Loss Curves

Creating a Function to Predict

Creating the function

```
def get_key(values):
    dictionary = {'joy': 0, 'anger': 1, 'love': 2, 'sadness': 3, 'fear': 4, 'surprise': 5}
    keys = []
    for value in values:
        max_index = np.argmax(value)
        for key, index in dictionary.items():
            if index == max_index:
                keys.append(key)
    return keys
```

```
[ ] def predict(sentence):
      sentence_lst=[]
      sentence_lst.append(sentence)
      sentence_seq=tokenizer.texts_to_sequences(sentence_lst)
      sentence_padded=pad_sequences(sentence_seq,maxlen=80,padding='post')
      ans=get_key(model.predict(sentence_padded))
      print("The emotion predicted is",ans)
```

We defined a function that takes input text and predicts the sentiment using the trained Bi-LSTM model.

OUTPUT

Checking our sentence

```
predict(str(input('Enter a sentence : ')))
```

```
Enter a sentence : i didnt feel humiliated
1/1 [==============================] - 0s 32ms/step
The emotion predicted is ['sadness']
```

```
predict(str(input('Enter a sentence : ')))
```

```
Enter a sentence : im grabbing a minute to post i feel greedy wrong
1/1 [==============================] - 0s 41ms/step
The emotion predicted is ['anger']
```

```
[ ] predict(str(input('Enter a sentence : ')))
```

```
Enter a sentence : i am ever feeling nostalgic about the fireplace i will know that it is still on the property
1/1 [==============================] - 0s 22ms/step
The emotion predicted is ['love']
```