

Name : ASHIKA C

Roll no : 225229105

```
In [1]: ▶ import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import nltk
import sklearn
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.optimizers import RMSprop, Adam
from keras.models import Sequential
from keras.layers import *
from nltk.corpus import stopwords
```

```
In [2]: ▶ data = pd.read_csv("news20.csv")
data.head()
```

Out[2]:

	category	text
0	tech	tv future in the hands of viewers with home th...
1	business	worldcom boss left books alone former worldc...
2	sport	tigers wary of farrell gamble leicester say ...
3	sport	yeading face newcastle in fa cup premiership s...
4	entertainment	ocean s twelve raids box office ocean s twelve...

```
In [3]: ▶ data.shape
```

Out[3]: (2225, 2)

```
In [4]: ▶ data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2225 entries, 0 to 2224
Data columns (total 2 columns):
#   Column      Non-Null Count  Dtype
---  -
0   category    2225 non-null   object
1   text        2225 non-null   object
dtypes: object(2)
memory usage: 34.9+ KB
```

```
In [5]: english_stops = set(stopwords.words('english'))
```

```
In [6]: y = data['category']
X=[]
for review in data['text']:
    filtered_sentence = [w.lower() for w in review.split() if not w in english_stops]
    X.append(filtered_sentence)
X = pd.Series(X)
```

```
In [7]: y_tokenizer = Tokenizer()
y_tokenizer.fit_on_texts(y)
y_seq = np.array(y_tokenizer.texts_to_sequences(y))
X_token = Tokenizer(num_words=5000, oov_token='<oov>')
X_token.fit_on_texts(X)
word_index = X_token.word_index
X_sequence = X_token.texts_to_sequences(X)
dict(list(word_index.items())[0:15])
```

```
Out[7]: {'<oov>': 1,
'said': 2,
'-': 3,
'mr': 4,
'would': 5,
'also': 6,
'people': 7,
'new': 8,
'us': 9,
'one': 10,
'could': 11,
'said.': 12,
'year': 13,
'last': 14,
'first': 15}
```

```
In [8]: X_padding= pad_sequences(X_sequence, maxlen=200, padding='post')
print(y_seq.shape)
print(X_padding.shape)
```

```
(2225, 1)
(2225, 200)
```

```
In [9]: x_train, x_test, y_train, y_test = train_test_split(X_padding, y_seq, test_size=0.2, random_state=42)
```

```
In [10]: ▸ vocab_size = 5000
embedding_dim = 64
max_length = 200
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(LSTM(embedding_dim))
model.add(Dense(embedding_dim, activation='tanh'))
model.add(Dense(6,activation='softmax'))
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 64)	320000
lstm (LSTM)	(None, 64)	33024
dense (Dense)	(None, 64)	4160
dense_1 (Dense)	(None, 6)	390
Total params: 357574 (1.36 MB)		
Trainable params: 357574 (1.36 MB)		
Non-trainable params: 0 (0.00 Byte)		

```
In [11]: ▶ model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',met  
history = model.fit(x_train,y_train, epochs=20, verbose=2, validation_spl
```

Epoch 1/20
45/45 - 14s - loss: 1.6435 - accuracy: 0.2514 - val_loss: 1.5901 - val_accuracy: 0.2444 - 14s/epoch - 316ms/step

Epoch 2/20
45/45 - 7s - loss: 1.4629 - accuracy: 0.3581 - val_loss: 1.3056 - val_accuracy: 0.4270 - 7s/epoch - 159ms/step

Epoch 3/20
45/45 - 8s - loss: 1.2197 - accuracy: 0.4066 - val_loss: 2.1563 - val_accuracy: 0.3118 - 8s/epoch - 175ms/step

Epoch 4/20
45/45 - 10s - loss: 1.6528 - accuracy: 0.3329 - val_loss: 1.3889 - val_accuracy: 0.3371 - 10s/epoch - 230ms/step

Epoch 5/20
45/45 - 7s - loss: 1.2053 - accuracy: 0.4424 - val_loss: 1.6691 - val_accuracy: 0.3202 - 7s/epoch - 151ms/step

Epoch 6/20
45/45 - 7s - loss: 1.1893 - accuracy: 0.4698 - val_loss: 1.1787 - val_accuracy: 0.4298 - 7s/epoch - 153ms/step

Epoch 7/20
45/45 - 7s - loss: 1.0962 - accuracy: 0.5414 - val_loss: 1.2625 - val_accuracy: 0.4242 - 7s/epoch - 154ms/step

Epoch 8/20
45/45 - 7s - loss: 1.0884 - accuracy: 0.5485 - val_loss: 1.8789 - val_accuracy: 0.3006 - 7s/epoch - 152ms/step

Epoch 9/20
45/45 - 7s - loss: 1.0505 - accuracy: 0.5035 - val_loss: 1.1410 - val_accuracy: 0.4410 - 7s/epoch - 153ms/step

Epoch 10/20
45/45 - 7s - loss: 1.1096 - accuracy: 0.5049 - val_loss: 1.3858 - val_accuracy: 0.4045 - 7s/epoch - 151ms/step

Epoch 11/20
45/45 - 7s - loss: 0.9143 - accuracy: 0.5801 - val_loss: 1.1253 - val_accuracy: 0.4747 - 7s/epoch - 157ms/step

Epoch 12/20
45/45 - 7s - loss: 0.8332 - accuracy: 0.6138 - val_loss: 1.0587 - val_accuracy: 0.4888 - 7s/epoch - 150ms/step

Epoch 13/20
45/45 - 7s - loss: 0.7721 - accuracy: 0.6419 - val_loss: 1.0822 - val_accuracy: 0.4803 - 7s/epoch - 150ms/step

Epoch 14/20
45/45 - 7s - loss: 0.7011 - accuracy: 0.6931 - val_loss: 1.1689 - val_accuracy: 0.5056 - 7s/epoch - 153ms/step

Epoch 15/20
45/45 - 7s - loss: 0.7657 - accuracy: 0.6369 - val_loss: 1.0378 - val_accuracy: 0.5225 - 7s/epoch - 150ms/step

Epoch 16/20
45/45 - 7s - loss: 0.7298 - accuracy: 0.6798 - val_loss: 1.2224 - val_accuracy: 0.4860 - 7s/epoch - 150ms/step

Epoch 17/20
45/45 - 7s - loss: 0.8023 - accuracy: 0.6728 - val_loss: 0.8225 - val_accuracy: 0.7219 - 7s/epoch - 152ms/step

Epoch 18/20
45/45 - 7s - loss: 0.7452 - accuracy: 0.7240 - val_loss: 1.0383 - val_accuracy: 0.5758 - 7s/epoch - 150ms/step

Epoch 19/20
45/45 - 7s - loss: 0.6798 - accuracy: 0.7416 - val_loss: 0.8665 - val_accuracy: 0.6713 - 7s/epoch - 154ms/step

Epoch 20/20

45/45 - 7s - loss: 0.5761 - accuracy: 0.7704 - val_loss: 1.0178 - val_accuracy: 0.6461 - 7s/epoch - 152ms/step

```
In [12]: ▶ model1 = Sequential()
model1.add(Embedding(vocab_size, embedding_dim))
model1.add(Conv1D(filters=32, kernel_size=5, strides=1, activation='relu'))
model1.add(MaxPooling1D((2)))
model1.add(LSTM(embedding_dim))
model1.add(Dense(128, activation='relu'))
model1.add(Dense(6, activation='softmax'))
model1.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
embedding_1 (Embedding)	(None, None, 64)	320000
conv1d (Conv1D)	(None, None, 32)	10272
max_pooling1d (MaxPooling1D)	(None, None, 32)	0
lstm_1 (LSTM)	(None, 64)	24832
dense_2 (Dense)	(None, 128)	8320
dense_3 (Dense)	(None, 6)	774
=====		
Total params: 364198 (1.39 MB)		
Trainable params: 364198 (1.39 MB)		
Non-trainable params: 0 (0.00 Byte)		

```
In [13]: ▶ model1.compile(optimizer='adam',loss='sparse_categorical_crossentropy',  
history1 = model1.fit(x_train,y_train, epochs=20,validation_split=0.2, ve
```

Epoch 1/20

45/45 - 12s - loss: 1.6424 - accuracy: 0.2563 - val_loss: 1.5689 - val_accuracy: 0.3343 - 12s/epoch - 262ms/step

Epoch 2/20

45/45 - 4s - loss: 1.3598 - accuracy: 0.3617 - val_loss: 1.2377 - val_accuracy: 0.4017 - 4s/epoch - 91ms/step

Epoch 3/20

45/45 - 5s - loss: 1.1110 - accuracy: 0.4171 - val_loss: 1.2128 - val_accuracy: 0.3399 - 5s/epoch - 102ms/step

Epoch 4/20

45/45 - 4s - loss: 1.0370 - accuracy: 0.4803 - val_loss: 0.9303 - val_accuracy: 0.5506 - 4s/epoch - 94ms/step

Epoch 5/20

45/45 - 4s - loss: 0.8156 - accuracy: 0.5730 - val_loss: 0.8693 - val_accuracy: 0.5871 - 4s/epoch - 94ms/step

Epoch 6/20

45/45 - 4s - loss: 0.7669 - accuracy: 0.5878 - val_loss: 0.9167 - val_accuracy: 0.5337 - 4s/epoch - 94ms/step

Epoch 7/20

45/45 - 4s - loss: 0.7271 - accuracy: 0.6124 - val_loss: 0.8524 - val_accuracy: 0.5871 - 4s/epoch - 91ms/step

Epoch 8/20

45/45 - 4s - loss: 0.6842 - accuracy: 0.6390 - val_loss: 0.8838 - val_accuracy: 0.5899 - 4s/epoch - 90ms/step

Epoch 9/20

45/45 - 4s - loss: 0.6064 - accuracy: 0.7058 - val_loss: 0.7325 - val_accuracy: 0.7163 - 4s/epoch - 89ms/step

Epoch 10/20

45/45 - 4s - loss: 0.3846 - accuracy: 0.8694 - val_loss: 1.3970 - val_accuracy: 0.5646 - 4s/epoch - 89ms/step

Epoch 11/20

45/45 - 4s - loss: 0.5864 - accuracy: 0.7542 - val_loss: 0.7597 - val_accuracy: 0.6938 - 4s/epoch - 88ms/step

Epoch 12/20

45/45 - 4s - loss: 0.5965 - accuracy: 0.7423 - val_loss: 0.9662 - val_accuracy: 0.5955 - 4s/epoch - 89ms/step

Epoch 13/20

45/45 - 4s - loss: 0.4547 - accuracy: 0.7746 - val_loss: 1.2960 - val_accuracy: 0.6601 - 4s/epoch - 88ms/step

Epoch 14/20

45/45 - 4s - loss: 0.3195 - accuracy: 0.9024 - val_loss: 0.4736 - val_accuracy: 0.8764 - 4s/epoch - 89ms/step

Epoch 15/20

45/45 - 4s - loss: 0.1583 - accuracy: 0.9572 - val_loss: 0.4431 - val_accuracy: 0.8904 - 4s/epoch - 89ms/step

Epoch 16/20

45/45 - 4s - loss: 0.0906 - accuracy: 0.9761 - val_loss: 0.5063 - val_accuracy: 0.8848 - 4s/epoch - 88ms/step

Epoch 17/20

45/45 - 5s - loss: 0.0713 - accuracy: 0.9874 - val_loss: 0.5558 - val_accuracy: 0.8708 - 5s/epoch - 101ms/step

Epoch 18/20

45/45 - 4s - loss: 0.0695 - accuracy: 0.9853 - val_loss: 0.4693 - val_accuracy: 0.8961 - 4s/epoch - 97ms/step

Epoch 19/20

45/45 - 4s - loss: 0.1698 - accuracy: 0.9670 - val_loss: 0.4143 - val_accuracy: 0.9017 - 4s/epoch - 91ms/step

Epoch 20/20

45/45 - 4s - loss: 0.1892 - accuracy: 0.9487 - val_loss: 0.5028 - val_accuracy: 0.8680 - 4s/epoch - 89ms/step

```
In [21]: ▶ from gensim.models import KeyedVectors
from gensim.scripts.glove2word2vec import glove2word2vec
```

```
In [22]: ▶ glove_file = "glove.6B.100d.txt"
glove_word2vec_file = "glove.6B.100d.txt.word2vec"
glove2word2vec(glove_file, glove_word2vec_file)
glove_embeddings = KeyedVectors.load_word2vec_format(glove_word2vec_file,
```

C:\Users\8mpira\AppData\Local\Temp\ipykernel_5460\889183730.py:3: DeprecationWarning: Call to deprecated `glove2word2vec` (KeyedVectors.load_word2vec_format(.., binary=False, no_header=True) loads GloVe text vector s.).

```
glove2word2vec(glove_file, glove_word2vec_file)
```

```
In [26]: ▶ embedding_matrix = np.zeros((vocab_size, 300))
for word, i in X_token.word_index.items():
    try:
        embedding_vector = glove_embeddings[word]
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector
    except:
        pass
```

```
In [ ]: ▶ model1 = Sequential()
model1.add(Embedding(vocab_size, embedding_dim))
model1.add(Conv1D(filters=32, kernel_size=5, strides=1, activation='relu'))
model1.add(MaxPooling1D((2)))
model1.add(LSTM(embedding_dim))
model1.add(Dense(128, activation='relu'))
model1.add(Dense(6, activation='softmax'))
model1.summary()
```

```
In [27]: model2 = Sequential()
model2.add(Embedding(vocab_size, 300, weights=[embedding_matrix], input_embeddings_dim=vocab_size, embedding_dim=300))
model2.add(Conv1D(filters=32, kernel_size=5, strides=1, activation='relu'))
model2.add(MaxPooling1D(2))
model2.add(LSTM(embedding_dim))
model2.add(Dense(128, activation='relu'))
model2.add(Dense(6, activation='softmax'))
model2.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 200, 300)	1500000
conv1d_1 (Conv1D)	(None, 196, 32)	48032
max_pooling1d_1 (MaxPooling1D)	(None, 98, 32)	0
lstm_2 (LSTM)	(None, 50)	16600
dense_4 (Dense)	(None, 128)	6528
dense_5 (Dense)	(None, 6)	774
Total params: 1571934 (6.00 MB)		
Trainable params: 71934 (280.99 KB)		
Non-trainable params: 1500000 (5.72 MB)		

```
In [30]: model2.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'],
history2 = model2.fit(x_train,y_train, epochs=5, verbose=2, validation_split=0.1))
```

```
Epoch 1/5
45/45 - 12s - loss: 1.6754 - accuracy: 0.2338 - val_loss: 1.6813 - val_accuracy: 0.2022 - 12s/epoch - 263ms/step
Epoch 2/5
45/45 - 5s - loss: 1.6716 - accuracy: 0.2338 - val_loss: 1.6780 - val_accuracy: 0.2022 - 5s/epoch - 116ms/step
Epoch 3/5
45/45 - 5s - loss: 1.6682 - accuracy: 0.2338 - val_loss: 1.6749 - val_accuracy: 0.2022 - 5s/epoch - 116ms/step
Epoch 4/5
45/45 - 5s - loss: 1.6650 - accuracy: 0.2338 - val_loss: 1.6721 - val_accuracy: 0.2022 - 5s/epoch - 115ms/step
Epoch 5/5
45/45 - 5s - loss: 1.6621 - accuracy: 0.2338 - val_loss: 1.6691 - val_accuracy: 0.2022 - 5s/epoch - 122ms/step
```

In []: ▶