In [53]:
```
##text-classification-with-bert-tokenizer-and-tf-2-0-in-python
```

In [54]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_re
mount=True).
```

In [55]:
```
# !pip install bert-for-tf2
```

In [56]:
```
#!pip install sentencepiece
```

In [57]:
```
#!mkdir ~/.kaggle
```

In [58]:
```
#!cp /kaggle.json ~/.kaggle/
```

In [59]:
```
#!chmod 600 ~/.kaggle/kaggle.json
```

In [60]:
```
#! pip install kaggle
```

In [61]:
```
#!pip install keras-tuner
```

In [62]:
```
#! kaggle datasets download -d sanjeetsinghnaik/quotes-from-goodread
```

In [63]:
```
#! unzip /content/quotes-from-goodread.zip
```

In [64]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [65]:
```
all_quotes = pd.read_csv("/content/all_quotes.csv")

all_quotes.isnull().values.any()

all_quotes.shape
```

Out[65]:
```
(30000, 5)
```

In [66]:
```
all_quotes.isnull().sum()
```

Out[66]:
```
Unnamed: 0       0
Quote            0
Author        1599
Main Tag         0
Other Tags       0
dtype: int64
```

In [67]:
```
all_quotes.columns
```

Out[67]:
```
Index(['Unnamed: 0', 'Quote', 'Author', 'Main Tag', 'Other Tags'], dtype='object')
```

In [68]:
```
all_quotes.head(2)
```

Out[68]:

	Unnamed: 0	Quote	Author	Main Tag	Other Tags
0	0	“Control of consciousness determines the quali...	— Mihaly Csikszentmihalyi, Flow: The Psycholog...	happiness	[' consciousness, happiness, quality-of-life']
1	1	“Copulation is no more foul to me than death is.”	— Walt Whitman, Leaves of Grass: The First (18...	death	[' death, sex']

In [69]:
```
all_quotes = all_quotes[["Quote", "Main Tag"]]
```

In [70]:
```
all_quotes.shape
```

Out[70]:
```
(30000, 2)
```

In [71]:
```
all_quotes.isnull().sum()
```

Out[71]:
```
Quote       0
Main Tag    0
dtype: int64
```

In [72]:
```
import re
```

In [73]:

```python
TAG_RE = re.compile(r'<[^>]+>')

def remove_tags(text):
    return TAG_RE.sub('', text)
```

In [74]:

```python
def preprocess_text(sen):
    # Removing html tags
    sentence = remove_tags(sen)

    # Remove punctuations and numbers
    sentence = re.sub('[^a-zA-Z]', ' ', sentence)

    # Single character removal
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)

    # Removing multiple spaces
    sentence = re.sub(r'\s+', ' ', sentence)

    return sentence
```

In [75]:

```python
import tensorflow as tf

import tensorflow_hub as hub

from tensorflow.keras import layers
import bert
```

In [76]:

```python
Quote = []
sentences = list(all_quotes['Quote'])
for sen in sentences:
    Quote.append(preprocess_text(sen))
```

In [77]:

```python
print(all_quotes.columns.values)
```

```
['Quote' 'Main Tag']
```

In [78]:

```python
all_quotes["Main Tag"].unique()
```

Out[78]:

```
array(['happiness', 'death', 'truth', 'poetry', 'inspiration', 'romance',
       'love', 'science', 'success', 'time'], dtype=object)
```

In [79]:

```python
all_quotes["Encoded Main Tag"] = all_quotes["Main Tag"].astype("category").cat.codes
```

In [80]:

```
all_quotes["Encoded Main Tag"].unique()
```

Out[80]:

```
array([1, 0, 9, 4, 2, 5, 3, 6, 7, 8], dtype=int8)
```

In [81]:

```
all_quotes.isnull().sum()
```

Out[81]:

```
Quote               0
Main Tag            0
Encoded Main Tag    0
dtype: int64
```

In [82]:

```
all_quotes.head(3)
```

Out[82]:

	Quote	Main Tag	Encoded Main Tag
0	“Control of consciousness determines the quali...	happiness	1
1	“Copulation is no more foul to me than death is.”	death	0
2	“Hope is a dream of which we long to have. Don...	happiness	1

In [83]:

```
data_texts = all_quotes["Quote"]
data_labels = all_quotes["Encoded Main Tag"]
```

In [84]:

```
data_texts.shape
```

Out[84]:

```
(30000,)
```

In [85]:

```
data_labels.shape
```

Out[85]:

```
(30000,)
```

In [86]:

```
import tensorflow as tf
```

In [87]:

```
tf.__version__
```

Out[87]:

```
'2.8.0'
```

In [88]:

```
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
```

In [89]:

```
### Vocabulary size
voc_size=5000
```

In [90]:

```
from sklearn.model_selection import train_test_split

# Split Train and Validation data
train_texts, val_texts, train_labels, val_labels = train_test_split(data_texts, data_labels, test_size=0.2, random_st
```

In [91]:

```
print(train_texts.shape,val_texts.shape)
print(train_labels.shape,val_labels.shape)
```

```
(24000,) (6000,)
(24000,) (6000,)
```

In [93]:

```
import tensorflow as tf
import tensorflow_hub as hub
#!pip install tensorflow-text
import tensorflow_text as text
```

In [95]:

```
bert_preprocess = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
bert_encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4")
```

In [96]:

```
# Bert layers
text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
preprocessed_text = bert_preprocess(text_input)
outputs = bert_encoder(preprocessed_text)
# Neural network layers
l = tf.keras.layers.Dropout(0.1, name="dropout")(outputs['pooled_output'])
l = tf.keras.layers.Dense(1, activation='sigmoid', name="output")(l)
# Use inputs and outputs to construct a final model
model = tf.keras.Model(inputs=[text_input], outputs = [l])
```

In [97]:

```
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 text (InputLayer)              [(None,)]            0           []                               
                                                                                                  
 keras_layer (KerasLayer)       {'input_mask': (Non  0           ['text[0][0]']                   
                                e, 128),                                                          
                                 'input_word_ids':                                                
                                (None, 128),                                                      
                                 'input_type_ids':                                                
                                (None, 128)}                                                      
                                                                                                  
 keras_layer_1 (KerasLayer)     {'default': (None,   109482241   ['keras_layer[0][0]',            
                                768),                             'keras_layer[0][1]',            
                                 'sequence_output':               'keras_layer[0][2]']            
                                 (None, 128, 768),                                                
                                 'pooled_output': (                                               
                                None, 768),                                                       
                                 'encoder_outputs':                                               
                                 [(None, 128, 768),                                               
                                 (None, 128, 768),                                                
                                 (None, 128, 768),                                                
                                 (None, 128, 768),                                                
                                 (None, 128, 768),                                                
                                 (None, 128, 768),                                                
                                 (None, 128, 768),                                                
                                 (None, 128, 768),                                                
                                 (None, 128, 768),                                                
                                 (None, 128, 768),                                                
                                 (None, 128, 768),                                                
                                 (None, 128, 768)]}                                               
                                                                                                  
 dropout (Dropout)              (None, 768)          0           ['keras_layer_1[0][13]']         
                                                                                                  
 output (Dense)                 (None, 1)            769         ['dropout[0][0]']                
                                                                                                  
==================================================================================================
Total params: 109,483,010
Trainable params: 769
Non-trainable params: 109,482,241
__________________________________________________________________________________________________
```

In [ ]:

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_texts, train_labels, epochs=20, batch_size = 32)
```

In [91]: