In [ ]:
```
!pip install flask-ngrok
```

```
Collecting flask-ngrok
  Downloading flask_ngrok-0.0.25-py3-none-any.whl (3.1 kB)
Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.7/dist-packages (from flask-ngrok) (1.1.4)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from flask-ngrok) (2.23.0)
Requirement already satisfied: Jinja2<3.0,>=2.10.1 in /usr/local/lib/python3.7/dist-packages (from Flask>=0.8->flask-ngrok) (2.11.3)
Requirement already satisfied: itsdangerous<2.0,>=0.24 in /usr/local/lib/python3.7/dist-packages (from Flask>=0.8->flask-ngrok) (1.1.0)
Requirement already satisfied: Werkzeug<2.0,>=0.15 in /usr/local/lib/python3.7/dist-packages (from Flask>=0.8->flask-ngrok) (1.0.1)
Requirement already satisfied: click<8.0,>=5.1 in /usr/local/lib/python3.7/dist-packages (from Flask>=0.8->flask-ngrok) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from Jinja2<3.0,>=2.10.1->Flask>=0.8->flask-ngrok) (2.0.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->flask-ngrok) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->flask-ngrok) (2021.10.8)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->flask-ngrok) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->flask-ngrok) (1.24.3)
Installing collected packages: flask-ngrok
Successfully installed flask-ngrok-0.0.25
```

In [ ]:
```
#importing libraries
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential,Model,load_model
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense,InputLayer,Embedding,Conv1D,MaxPooling1D,Input,LSTM,BatchNormalization,Bidirectional
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras import regularizers
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping,Callback
from flask_ngrok import run_with_ngrok
from flask import Flask,render_template,jsonify,request
```

In [ ]:
```
!pip install transformers
```

```
Collecting transformers
  Downloading transformers-4.19.2-py3-none-any.whl (4.2 MB)
     |████████████████████████████████| 4.2 MB 4.1 MB/s
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (21.3)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.23.0)
Collecting tokenizers!=0.11.3,<0.13,>=0.11.1
  Downloading tokenizers-0.12.1-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (6.6 MB)
     |████████████████████████████████| 6.6 MB 13.1 MB/s
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from transformers) (4.11.3)
Collecting pyyaml>=5.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (596 kB)
     |████████████████████████████████| 596 kB 40.7 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.7.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (1.21.6)
Collecting huggingface-hub<1.0,>=0.1.0
  Downloading huggingface_hub-0.6.0-py3-none-any.whl (84 kB)
     |████████████████████████████████| 84 kB 3.0 MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (2019.12.20)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.64.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from huggingface-hub<1.0,>=0.1.0->transformers) (4.2.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=20.0->transformers) (3.0.9)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->transformers) (3.8.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (2021.10.8)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (2.10)
Installing collected packages: pyyaml, tokenizers, huggingface-hub, transformers
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed huggingface-hub-0.6.0 pyyaml-6.0 tokenizers-0.12.1 transformers-4.19.2
```

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

```
Mounted at /content/drive
```

In [ ]:
```
# from transformers import BertTokenizer, TFBertForQuestionAnswering
import tensorflow as tf
from transformers import  RobertaConfig, TFRobertaForQuestionAnswering
from transformers import RobertaTokenizer
tokenizer = RobertaTokenizer.from_pretrained("ydshieh/roberta-base-squad2")
Robert_model = TFRobertaForQuestionAnswering.from_pretrained("ydshieh/roberta-base-squad2")
```

```
All model checkpoint layers were used when initializing TFRobertaForQuestionAnswering.

All the layers of TFRobertaForQuestionAnswering were initialized from the model checkpoint at ydshieh/roberta-base-squad2.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFRobertaForQuestionAnswering for predictions without further training.
```

In [ ]:
```
#Load and read train csv data
train_data=pd.read_csv('/content/train_preprocess.csv')
```

In [ ]:
```
X_train,X_test,y_train,y_test=train_test_split(train_data,train_data.sentiment,test_size=0.15,stratify=train_data.sentiment)
```

In [ ]:
```
X_train.dropna(inplace=True)
X_test.dropna(inplace=True)
y_train.dropna(inplace=True)
y_test.dropna(inplace=True)
# remove own index with default index
X_train.reset_index(inplace = True, drop = True)
X_test.reset_index(inplace = True, drop = True)
y_train.reset_index(inplace = True, drop = True)
y_test.reset_index(inplace = True, drop = True)
```

In [ ]:
```
MAX_LEN=102
def data_input(df):
    count = df.shape[0]
    input_ids = np.zeros((count,MAX_LEN),dtype='int32')
    attention_masks = np.zeros((count,MAX_LEN),dtype='int32')
    token_type_ids = np.zeros((count,MAX_LEN),dtype='int32')
    for i in range(count):
      encoded = tokenizer.encode_plus(
          df.sentiment[i],
          df.text[i],
          add_special_tokens=True,
          max_length=MAX_LEN,
          pad_to_max_length=True,
          return_token_type_ids=True,
          return_attention_mask=True, truncation=True,return_tensors='tf'
      )
      input_ids[i] =encoded['input_ids']
      attention_masks[i] =encoded['attention_mask']
    return input_ids,attention_masks
```

In [ ]:
```
train_input_ids,train_attention_masks=data_input(X_train)#train
test_input_ids,test_attention_masks=data_input(X_test)#test
```

```
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned lists will always be empty even if some tokens have been removed.
```

In [ ]:
```
#The last thing that we have to do before we begin training the model is to convert the selected text column as a combination of the START and END tokens.
#https://www.kaggle.com/code/parulpandey/eda-and-preprocessing-for-bert/comments
def Get_Start_End_tokes(df):

  count = df.shape[0]
  start_tokens = np.zeros((count,MAX_LEN),dtype='int32')
  end_tokens = np.zeros((count,MAX_LEN),dtype='int32')
  for k in range(0,count):
      text_t = " "+" ".join(df.loc[k,'text'].split())
      text_s = " "+" ".join(df.loc[k,'selected_text'].split())
      #print(text_t)
      #print(text_s)
      idx = text_t.find(text_s)
      #print(idx)
      chars = np.zeros((len(text_t)))
      chars[idx:idx+len(text_s)]=1

      if text_t[idx-1]==' ':
          chars[idx-1] = 1
      #print(chars)
      #print("length of chars:",len(chars))
      offsets = []; idx=0
      enc = tokenizer.encode(text_t)
      for t in enc:
          w = tokenizer.decode([t])
          #print(w,t)
          if t==0 or t==2:
              offsets.append((idx,idx+0))
              idx +=0
          else:
              offsets.append((idx,idx+len(w)))
              idx += len(w)
      #print(offsets)
      toks = []
      s= np.argmax(chars)
      e= len(chars)-1-np.argmax(chars[::-1])
      #print("common:",s,e)
      for i,(a,b) in enumerate(offsets):
          sm = np.sum(chars[a:b])
          if sm>0:
              toks.append(i)

      if len(toks)>0:

          start_tokens[k,toks[0]] = 1
          end_tokens[k,toks[-1]+1] = 1


  return start_tokens,end_tokens
```

In [ ]:
```
train_start_tokens,train_end_tokens=Get_Start_End_tokes(X_train)#train
test_start_tokens,test_end_tokens=Get_Start_End_tokes(X_test)#test
```

In [ ]:
```
input_data=[train_input_ids ,train_attention_masks]
output_data = [train_start_tokens,train_end_tokens]

test_input =[test_input_ids,test_attention_masks]
test_output =[test_start_tokens,test_end_tokens]
```

In [ ]:
```
#Finds the selected text for the given tweet
def find_selected_text(data,tokenizer,start,end):
  '''Finds the selected text for the given tweet'''
  selected_text_list=[]
  for i in range(data.shape[0]):

      #Finding the start and end index
      start_idx=np.argmax(start[i])
      end_idx=np.argmax(end[i])


      #If start is greater than end index, predicted_text=text
      if (start_idx>end_idx):
          predicted_text=data.loc[i,'text']
          selected_text_list.append(predicted_text)

      else:
          text1 = " "+" ".join(data.loc[i,'text'].split())
          enc=tokenizer.encode(text1)
          #predicted_text=tokenizer.decode(tokens.ids[start_idx-1:end_idx])
          predicted_text= tokenizer.decode(enc[start_idx:end_idx+1])
          #print(tokenizer.decode(enc[toks[0]:toks[-1]+1]))
          selected_text_list.append(predicted_text)
  return selected_text_list
```

In [ ]:
```
#Model Archietecture
MAX_LEN=102
ids=Input((MAX_LEN),name='ids',dtype='int32')
att_mask=Input((MAX_LEN),name='att_mask',dtype='int32')

bert_output=Robert_model([ids,att_mask])

dropout1=Dropout(0.2,name='dropout1')(bert_output[0])
out_1 = tf.keras.layers.Activation('softmax',name='activation1')(dropout1)

dropout2=Dropout(0.2,name='dropout2')(bert_output[1])
out_2 = tf.keras.layers.Activation('softmax',name='activation2')(dropout2)

history_res = Model(inputs=[ids, att_mask], outputs=[out_1,out_2])
optimizer = tf.keras.optimizers.Adam(learning_rate=3e-5)
    # model.compile(loss='binary_crossentropy', optimizer=optimizer)
history_res.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

In [ ]:
```
history_res.load_weights('/content/drive/My Drive/Tweet_BertResult_model/weights_Updated_new.best.hdf5') #Load the save weigted
```

In [ ]:
```
start_pred,end_pred = history_res.predict((input_data))
```

In [ ]:
```
val= {'text':["i hate this dish"],'sentiment':["negative"]}

df = pd.DataFrame(val)
print(df)
selected_text_1=find_selected_text(df,tokenizer,start_pred,end_pred)

print(selected_text_1)
```

```
              text sentiment
0  i hate this dish  negative
[' i hate']
```

In [74]:
```
import random

app = Flask(__name__,template_folder='/content/drive/MyDrive/templates')
run_with_ngrok(app)

@app.route("/")
def Index():
    return render_template('index.html')
@app.route('/predict', methods=['GET','POST'])
def predict():
        vals = [each for each in request.form.values()]
        values = {'text':[vals[0]],'sentiment':[vals[1]]}
        df = pd.DataFrame(values)
        output =find_selected_text(df,tokenizer,start_pred,end_pred)
        return render_template('index.html',text=vals[0],prediction=output,sentiment=vals[1])

if __name__ == "__main__":
    port = 5000 + random.randint(0, 999)

    #url = "http://127.0.0.1:{0}".format(port)
    app.run()
  # app.run(use_reloader=False, port=port)
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Running on http://2c13-35-236-132-249.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
```