SEA Team

Twitter sentiment Extraction

Twitter sentiment Extraction

- In this competition you will need to pick out the part of the tweet (word or phrase) that reflects the sentiment.
- What words in tweets support a positive, negative, or neutral sentiment? How can we help make that determination using machine learning tools?
- Our objective in this competition is to construct a model that can look at the labeled sentiment for a given tweet and figure out what word or phrase best supports it.

Primitive model “using spacy“

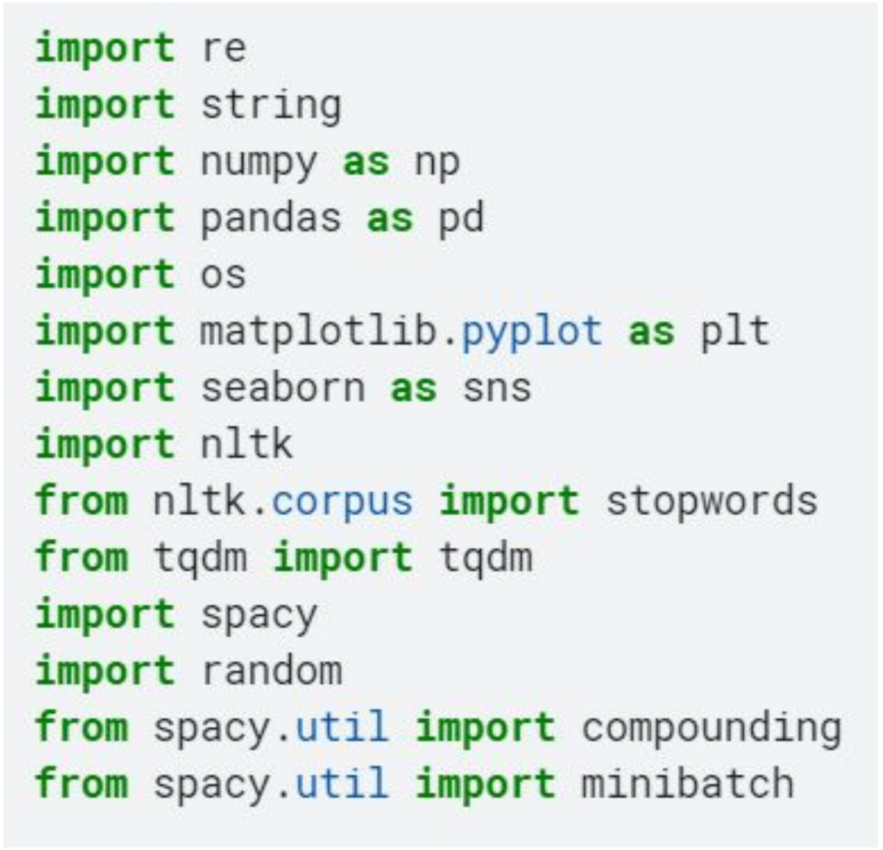
- NER Problem
- Importing Necessities
- Reading data
- Data visualization
- Build model
- Train model
- Run test

NER problem :

- Named-entity recognition (NER) is a subtask of information extraction that seeks to locate and classify named entity mentioned in unstructured text into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

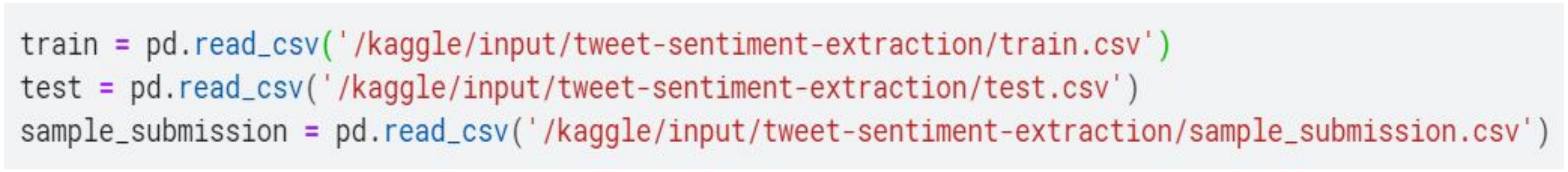
Importing Necessities

★ Import spacy library.

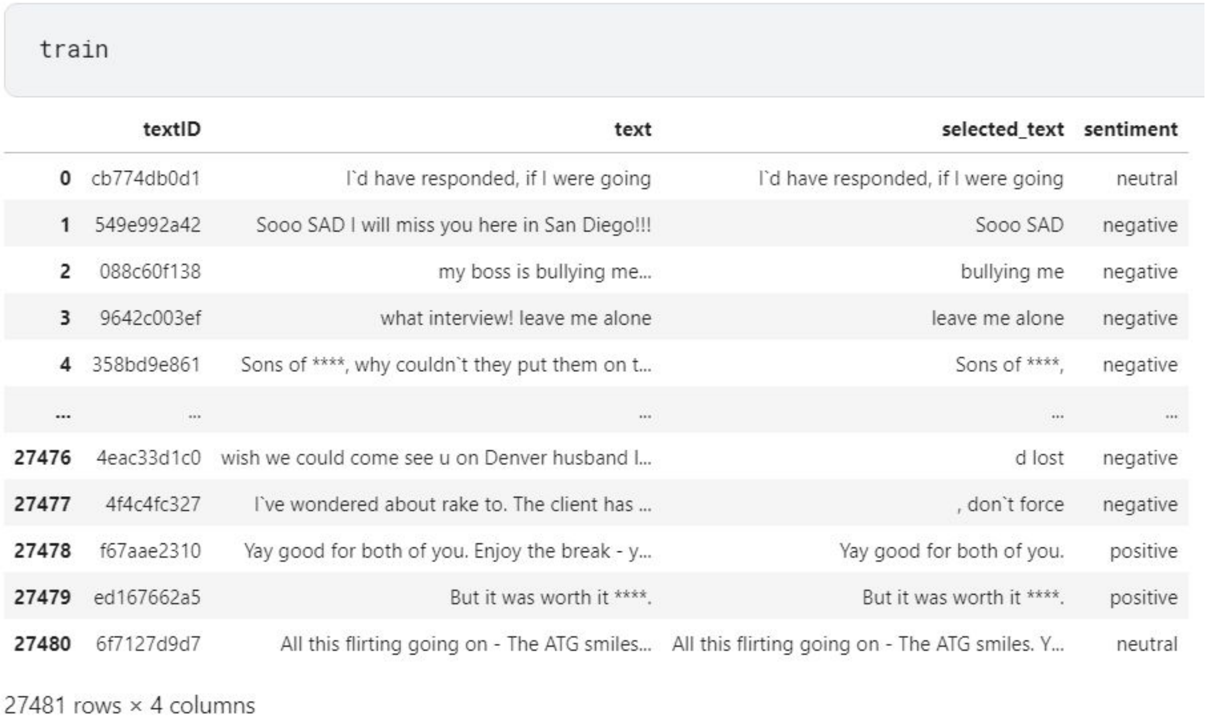


```
import re
import string
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from tqdm import tqdm
import spacy
import random
from spacy.util import compounding
from spacy.util import minibatch
```

Reading data



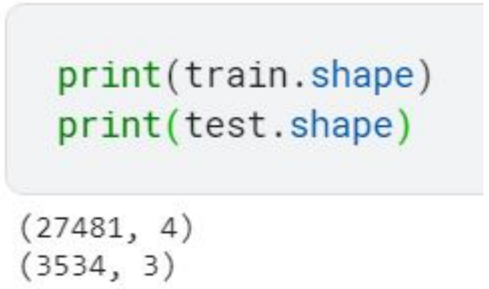
```python
train = pd.read_csv('/kaggle/input/tweet-sentiment-extraction/train.csv')
test = pd.read_csv('/kaggle/input/tweet-sentiment-extraction/test.csv')
sample_submission = pd.read_csv('/kaggle/input/tweet-sentiment-extraction/sample_submission.csv')
```



```python
train
```

	textID	text	selected_text	sentiment
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative
...	...	...	...	...
27476	4eac33d1c0	wish we could come see u on Denver husband l...	d lost	negative
27477	4f4c4fc327	I`ve wondered about rake to. The client has ...	, don`t force	negative
27478	f67aae2310	Yay good for both of you. Enjoy the break - y...	Yay good for both of you.	positive
27479	ed167662a5	But it was worth it ****.	But it was worth it ****.	positive
27480	6f7127d9d7	All this flirting going on - The ATG smiles...	All this flirting going on - The ATG smiles. Y...	neutral

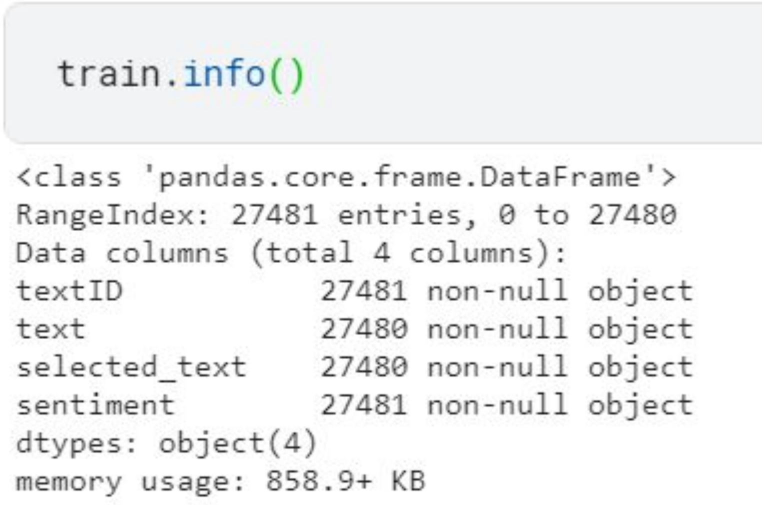
27481 rows × 4 columns



```python
print(train.shape)
print(test.shape)
```

```
(27481, 4)
(3534, 3)
```

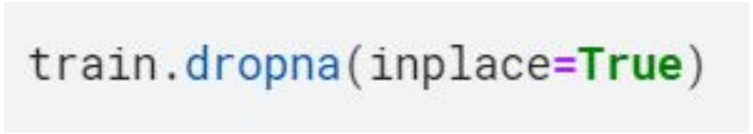
Data Visualization



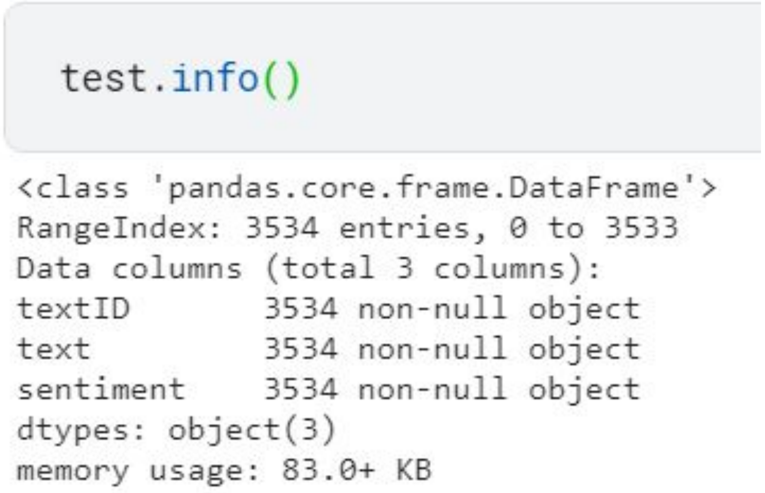
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27481 entries, 0 to 27480
Data columns (total 4 columns):
textID           27481 non-null object
text             27480 non-null object
selected_text    27480 non-null object
sentiment        27481 non-null object
dtypes: object(4)
memory usage: 858.9+ KB
```

Null rows should be dropped



```
train.dropna(inplace=True)
```



```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3534 entries, 0 to 3533
Data columns (total 3 columns):
textID       3534 non-null object
text         3534 non-null object
sentiment    3534 non-null object
dtypes: object(3)
memory usage: 83.0+ KB
```

No Null Rows

Jaccard Coefficient :

- ❑ Implementing Jaccard by calculating intersection over union .

```
def jaccard (text1, text2):
    a = set(text1.split())
    b = set(text2.split())
    intresection = a.intersection(b)
    IOU =(float) (len(intresection))/(len(a)+len(b)-len(intresection))
    return IOU
```

- ❑ applying jaccard between text and selected text.

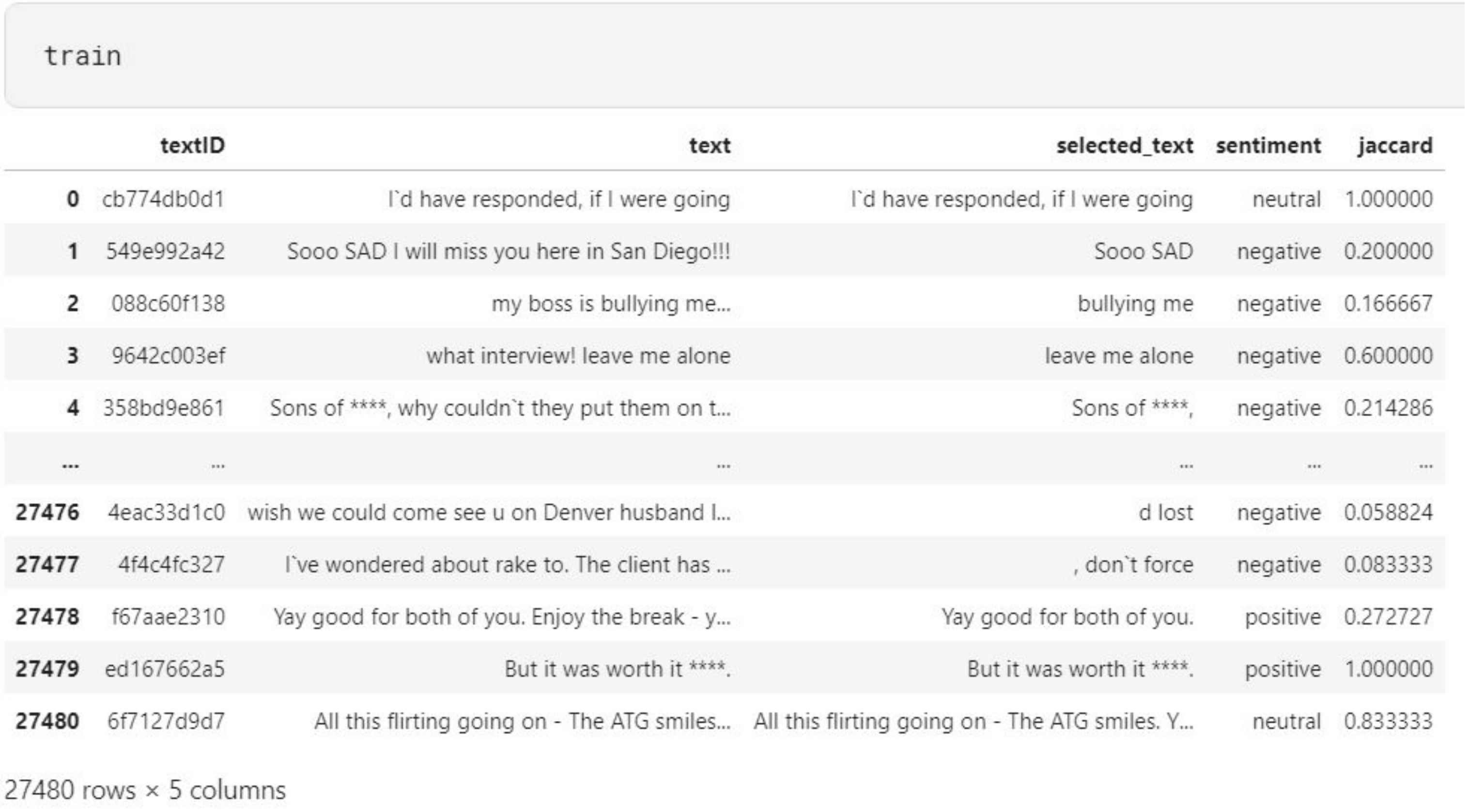
```
jaccard_list=[]
def calc_jaccard():
    for row in train.itertuples():
        jaccard_list.append(jaccard (row.text, row.selected_text))
    return jaccard_list;
```

- ❑ Calculating jaccard and inserting it as column in train data .

```
jac = calc_jaccard()
```

```
train['jaccard'] = jac
```

Jaccard Coefficient :



```
train
```

	textID	text	selected_text	sentiment	jaccard
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral	1.000000
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative	0.200000
2	088c60f138	my boss is bullying me...	bullying me	negative	0.166667
3	9642c003ef	what interview! leave me alone	leave me alone	negative	0.600000
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative	0.214286
...	...	...	...	...	...
27476	4eac33d1c0	wish we could come see u on Denver husband l...	d lost	negative	0.058824
27477	4f4c4fc327	I`ve wondered about rake to. The client has ...	, don`t force	negative	0.083333
27478	f67aae2310	Yay good for both of you. Enjoy the break - y...	Yay good for both of you.	positive	0.272727
27479	ed167662a5	But it was worth it ****.	But it was worth it ****.	positive	1.000000
27480	6f7127d9d7	All this flirting going on - The ATG smiles...	All this flirting going on - The ATG smiles. Y...	neutral	0.833333

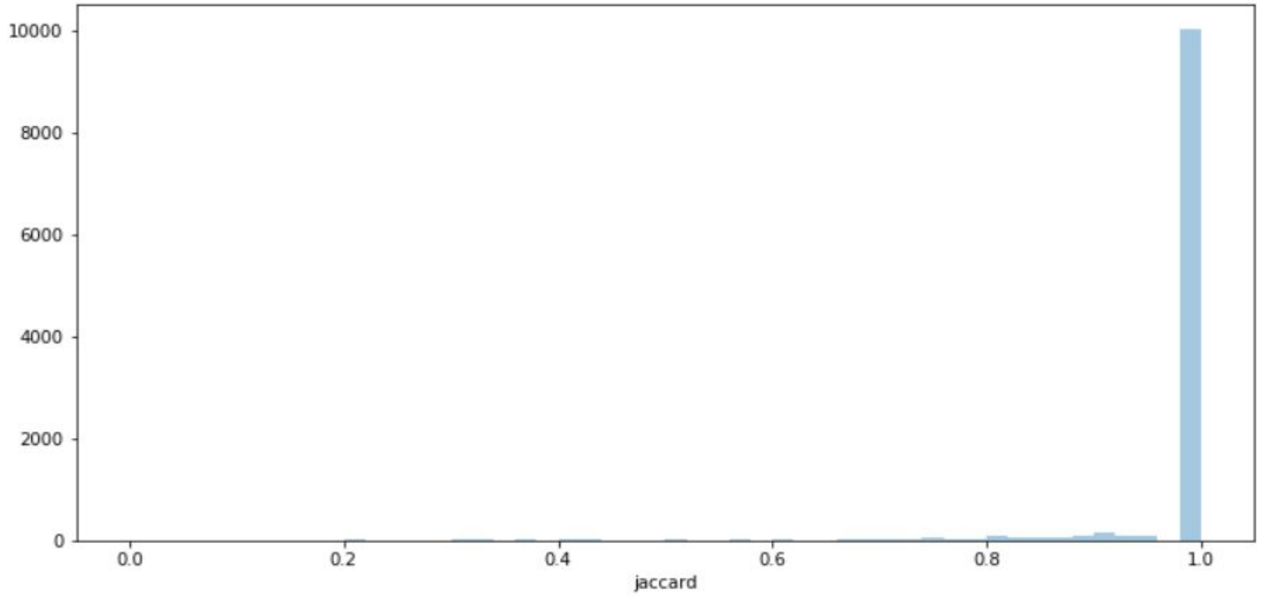
27480 rows × 5 columns

Jaccard Coefficient :

- Plotting jaccard of neural tweets .

```
plt.figure(figsize=(12,6))
sns.distplot(train[train['sentiment']=='neutral']['jaccard'],kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ed85a7710>
```

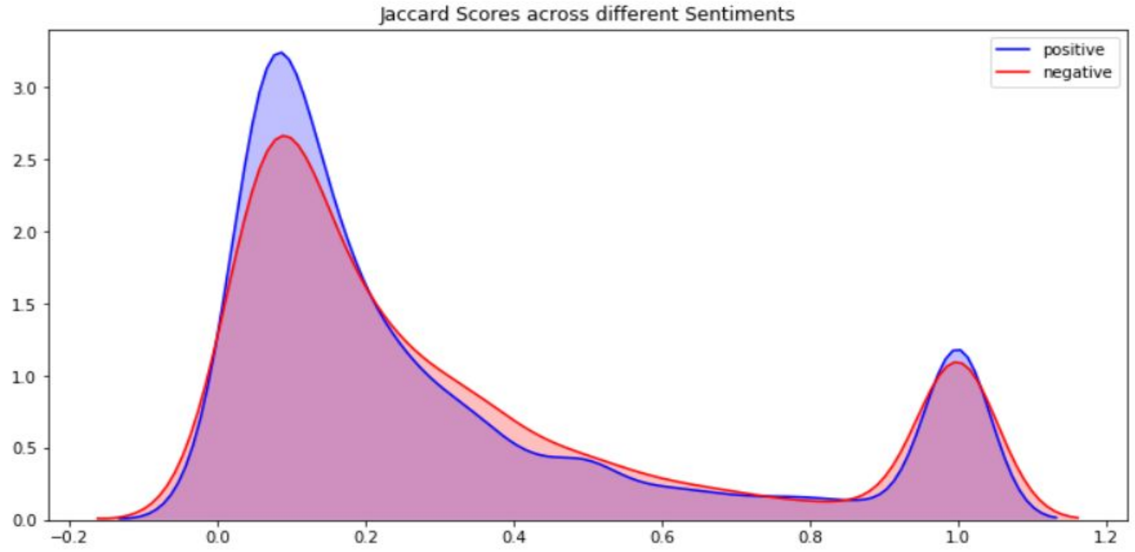


Jaccard Coefficient :

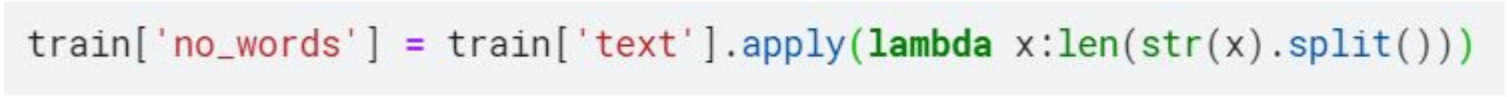
- ❑ Plotting jaccard of positive and negative tweets .

```
plt.figure(figsize=(12,6))
p1=sns.kdeplot(train[train['sentiment']=='positive']['jaccard'], shade=True, color="b").set_title('Jaccard Scores across diff
p2=sns.kdeplot(train[train['sentiment']=='negative']['jaccard'], shade=True, color="r")
plt.legend(labels=['positive','negative'])
```

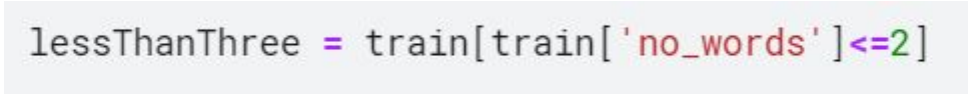
```
<matplotlib.legend.Legend at 0x7f9ed8399198>
```



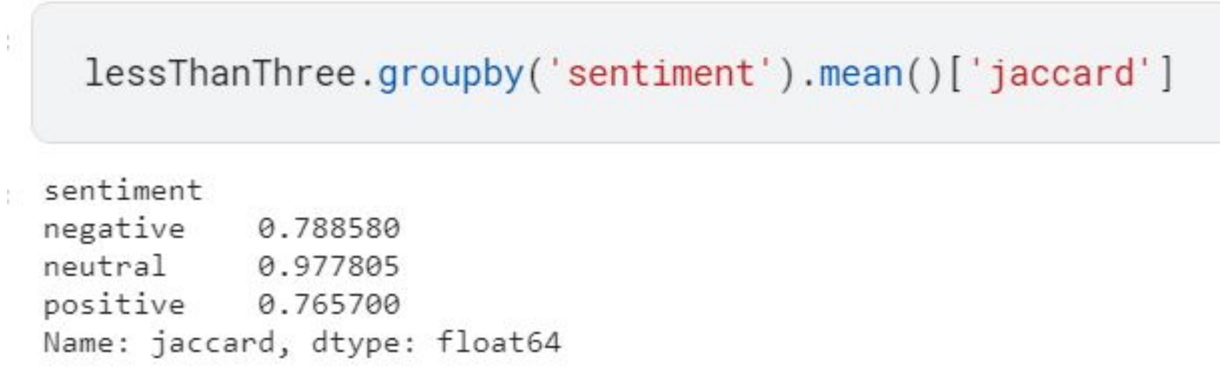
Jaccard Coefficient :



```
train['no_words'] = train['text'].apply(lambda x:len(str(x).split()))
```



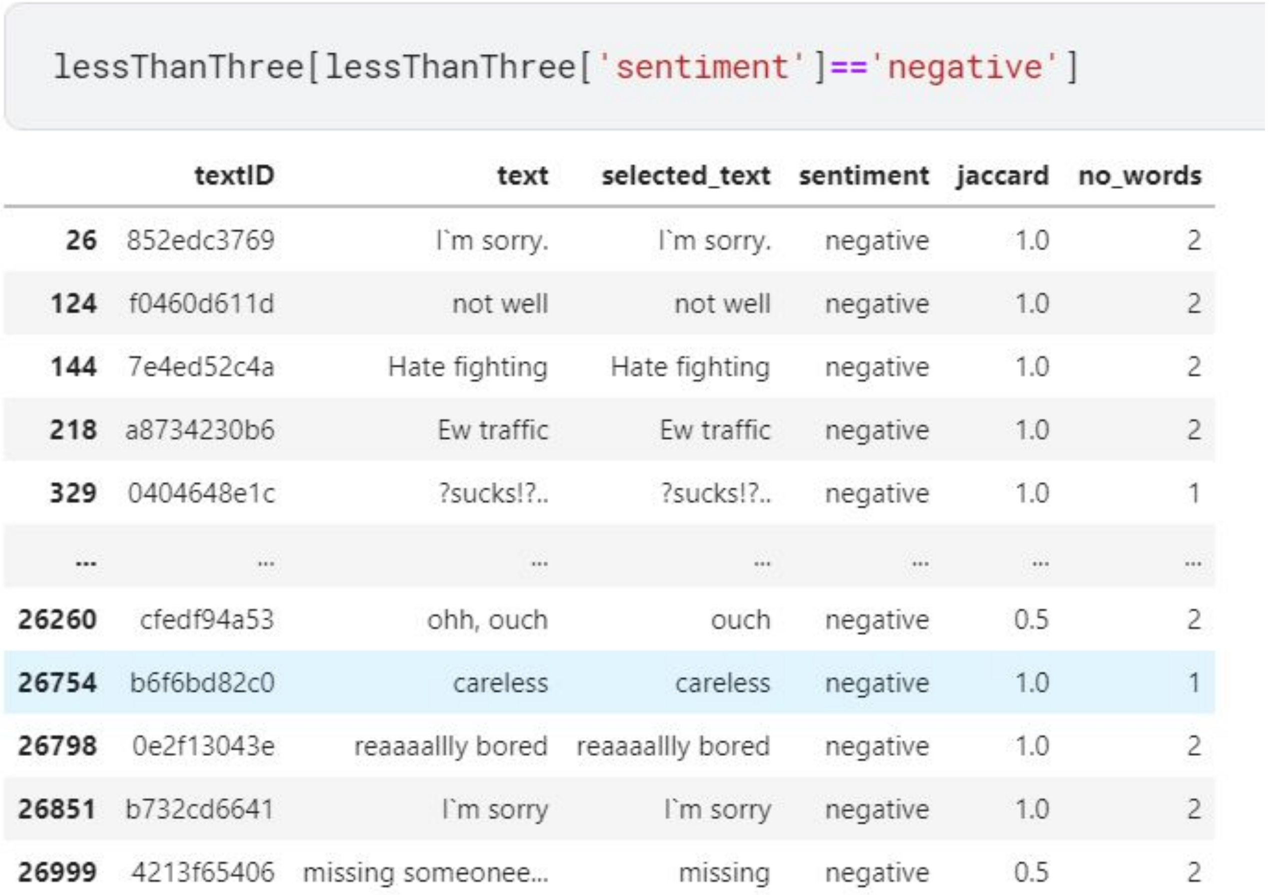
```
lessThanThree = train[train['no_words']<=2]
```



```
lessThanThree.groupby('sentiment').mean()['jaccard']
```

```
sentiment
negative    0.788580
neutral     0.977805
positive    0.765700
Name: jaccard, dtype: float64
```

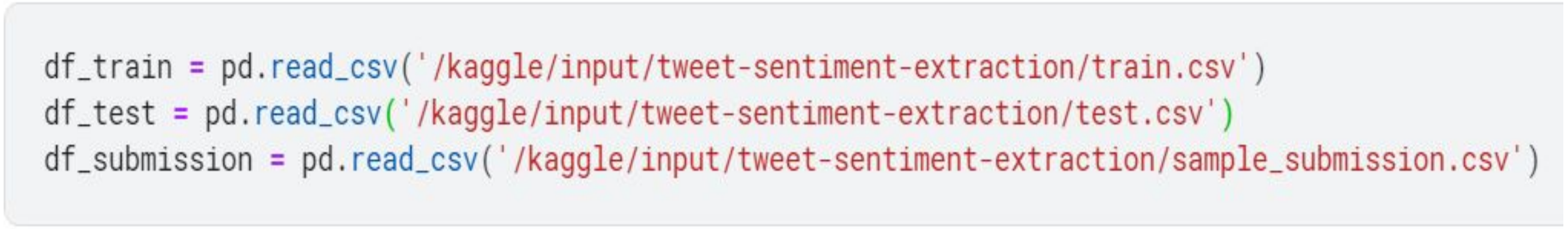
Jaccard Coefficient :



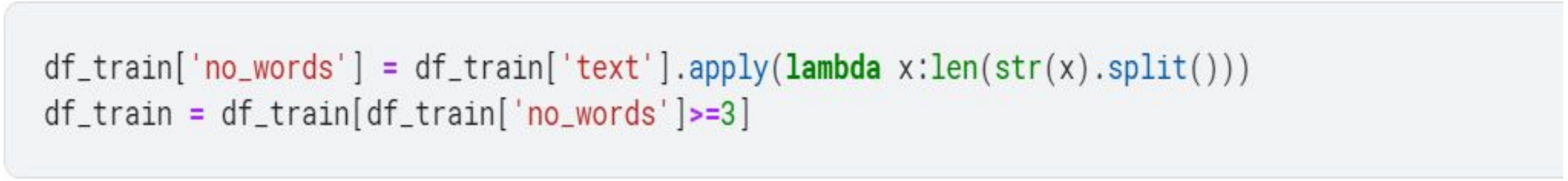
```
lessThanThree[lessThanThree['sentiment']=='negative']
```

	textID	text	selected_text	sentiment	jaccard	no_words
26	852edc3769	I`m sorry.	I`m sorry.	negative	1.0	2
124	f0460d611d	not well	not well	negative	1.0	2
144	7e4ed52c4a	Hate fighting	Hate fighting	negative	1.0	2
218	a8734230b6	Ew traffic	Ew traffic	negative	1.0	2
329	0404648e1c	?sucks!?..	?sucks!?..	negative	1.0	1
...	...	...	...	...	...	...
26260	cfedf94a53	ohh, ouch	ouch	negative	0.5	2
26754	b6f6bd82c0	careless	careless	negative	1.0	1
26798	0e2f13043e	reaaaallly bored	reaaaallly bored	negative	1.0	2
26851	b732cd6641	I`m sorry	I`m sorry	negative	1.0	2
26999	4213f65406	missing someonee...	missing	negative	0.5	2

Train data with number of words greater than 2 :

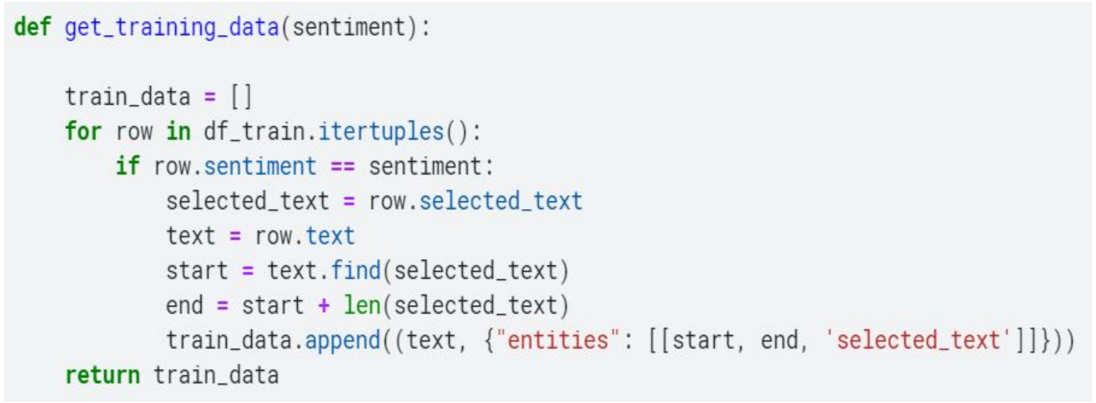


```
df_train = pd.read_csv('/kaggle/input/tweet-sentiment-extraction/train.csv')
df_test = pd.read_csv('/kaggle/input/tweet-sentiment-extraction/test.csv')
df_submission = pd.read_csv('/kaggle/input/tweet-sentiment-extraction/sample_submission.csv')
```

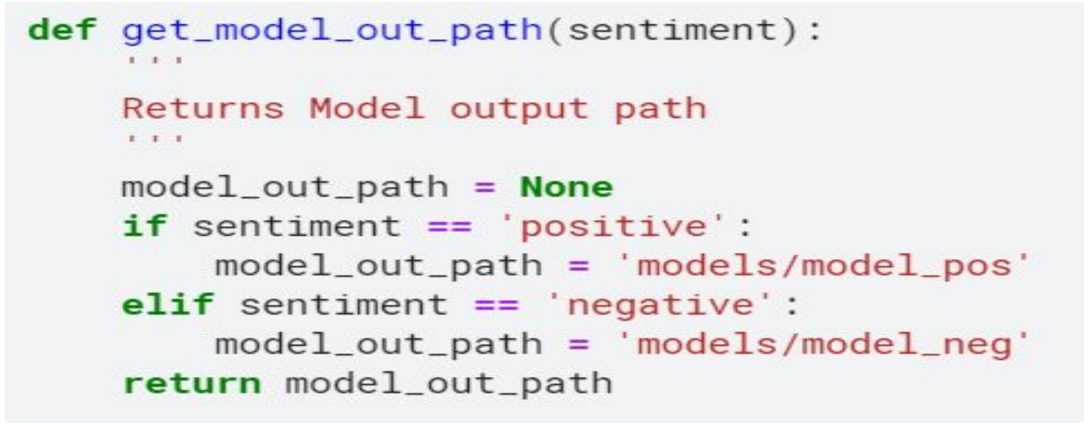


```
df_train['no_words'] = df_train['text'].apply(lambda x:len(str(x).split()))
df_train = df_train[df_train['no_words']>=3]
```

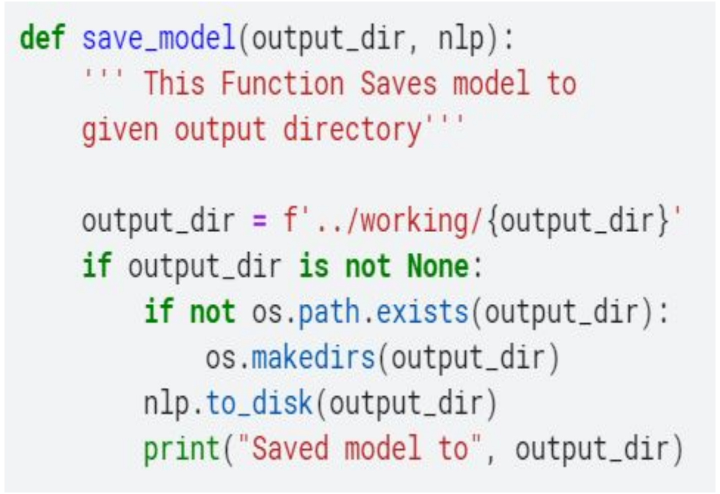
Helping Functions :



```python
def get_training_data(sentiment):

    train_data = []
    for row in df_train.itertuples():
        if row.sentiment == sentiment:
            selected_text = row.selected_text
            text = row.text
            start = text.find(selected_text)
            end = start + len(selected_text)
            train_data.append((text, {"entities": [[start, end, 'selected_text']]}))
    return train_data
```

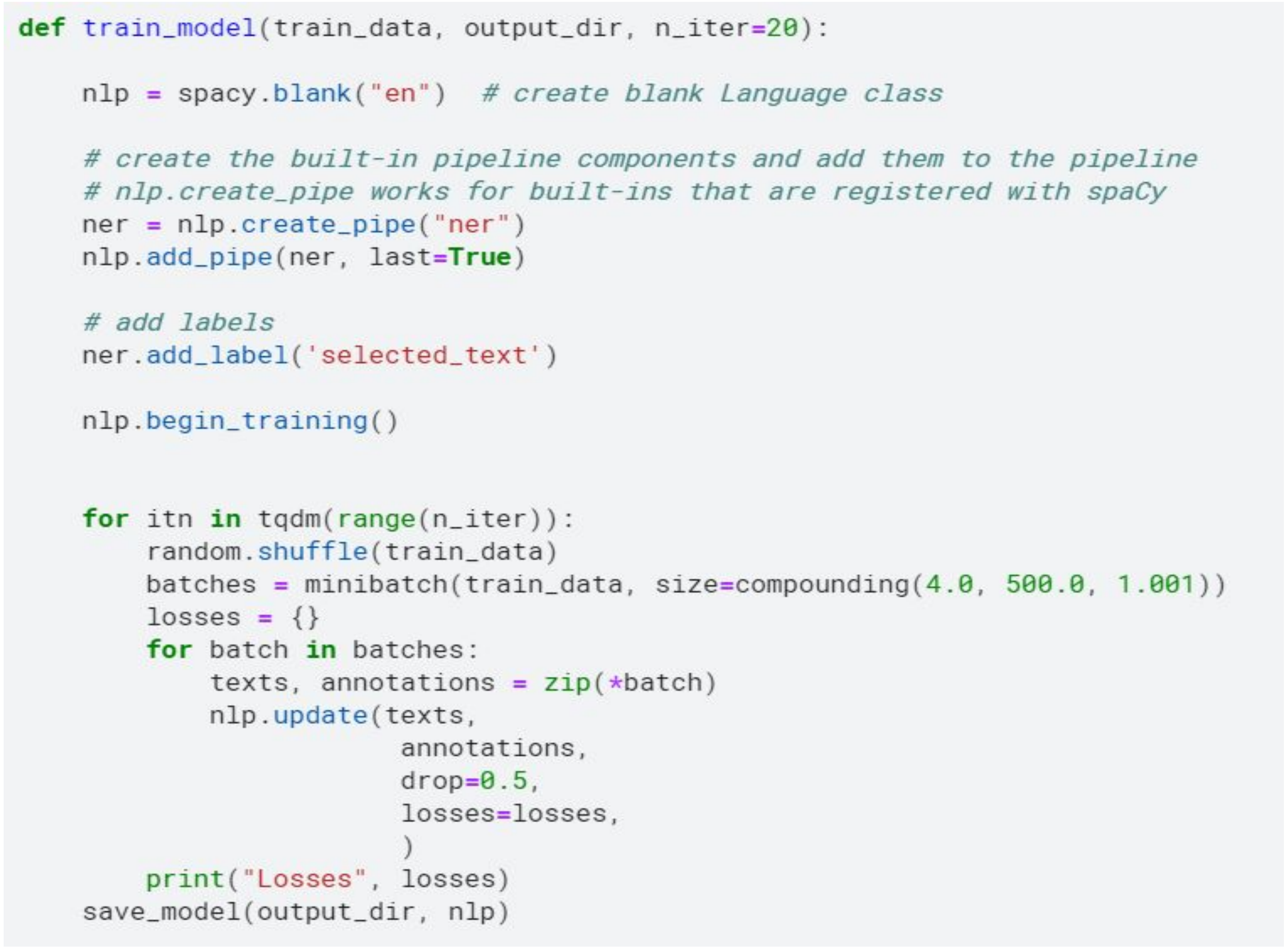


```python
def get_model_out_path(sentiment):
    '''
    Returns Model output path
    '''
    model_out_path = None
    if sentiment == 'positive':
        model_out_path = 'models/model_pos'
    elif sentiment == 'negative':
        model_out_path = 'models/model_neg'
    return model_out_path
```



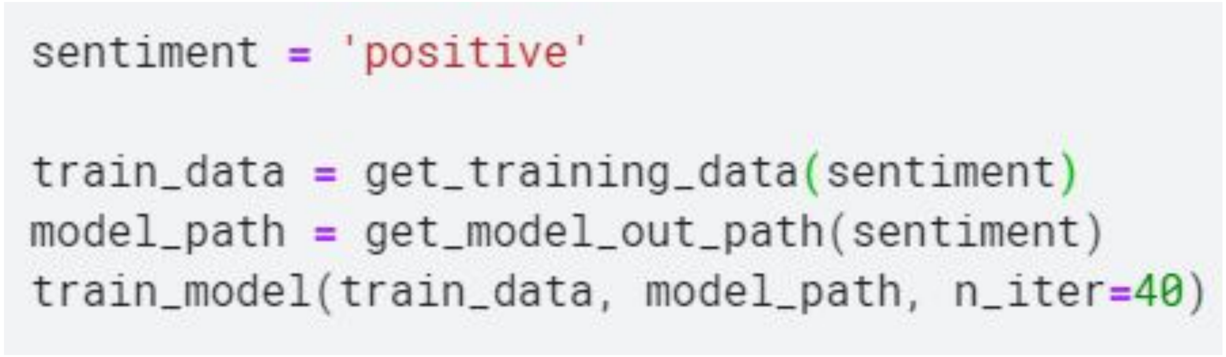
```python
def save_model(output_dir, nlp):
    ''' This Function Saves model to
    given output directory'''

    output_dir = f'../working/{output_dir}'
    if output_dir is not None:
        if not os.path.exists(output_dir):
            os.makedirs(output_dir)
        nlp.to_disk(output_dir)
        print("Saved model to", output_dir)
```

Train model :

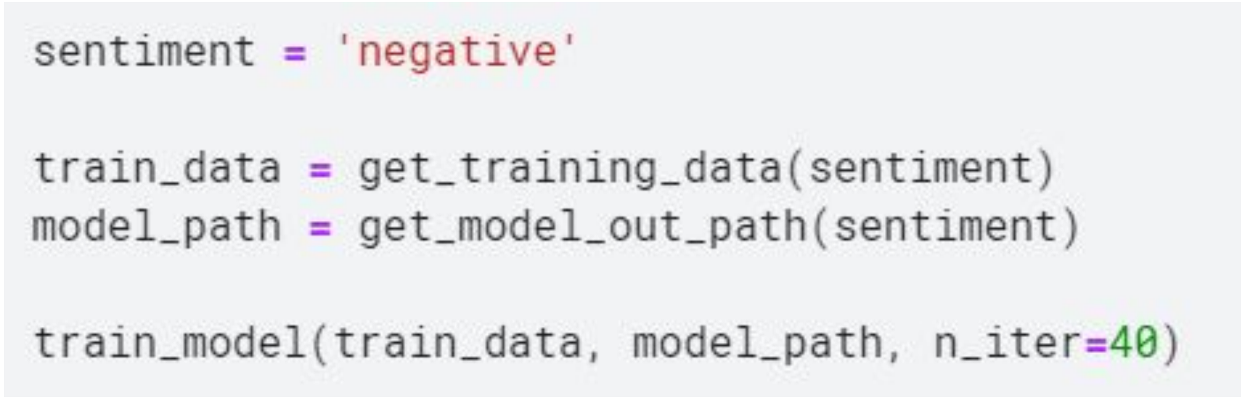


```python
def train_model(train_data, output_dir, n_iter=20):

    nlp = spacy.blank("en")  # create blank Language class

    # create the built-in pipeline components and add them to the pipeline
    # nlp.create_pipe works for built-ins that are registered with spaCy
    ner = nlp.create_pipe("ner")
    nlp.add_pipe(ner, last=True)

    # add labels
    ner.add_label('selected_text')

    nlp.begin_training()


    for itn in tqdm(range(n_iter)):
        random.shuffle(train_data)
        batches = minibatch(train_data, size=compounding(4.0, 500.0, 1.001))
        losses = {}
        for batch in batches:
            texts, annotations = zip(*batch)
            nlp.update(texts,
                        annotations,
                        drop=0.5,
                        losses=losses,
                        )
        print("Losses", losses)
    save_model(output_dir, nlp)
```

Implementing 2 models (positive/negative) with number of iterations 40 :

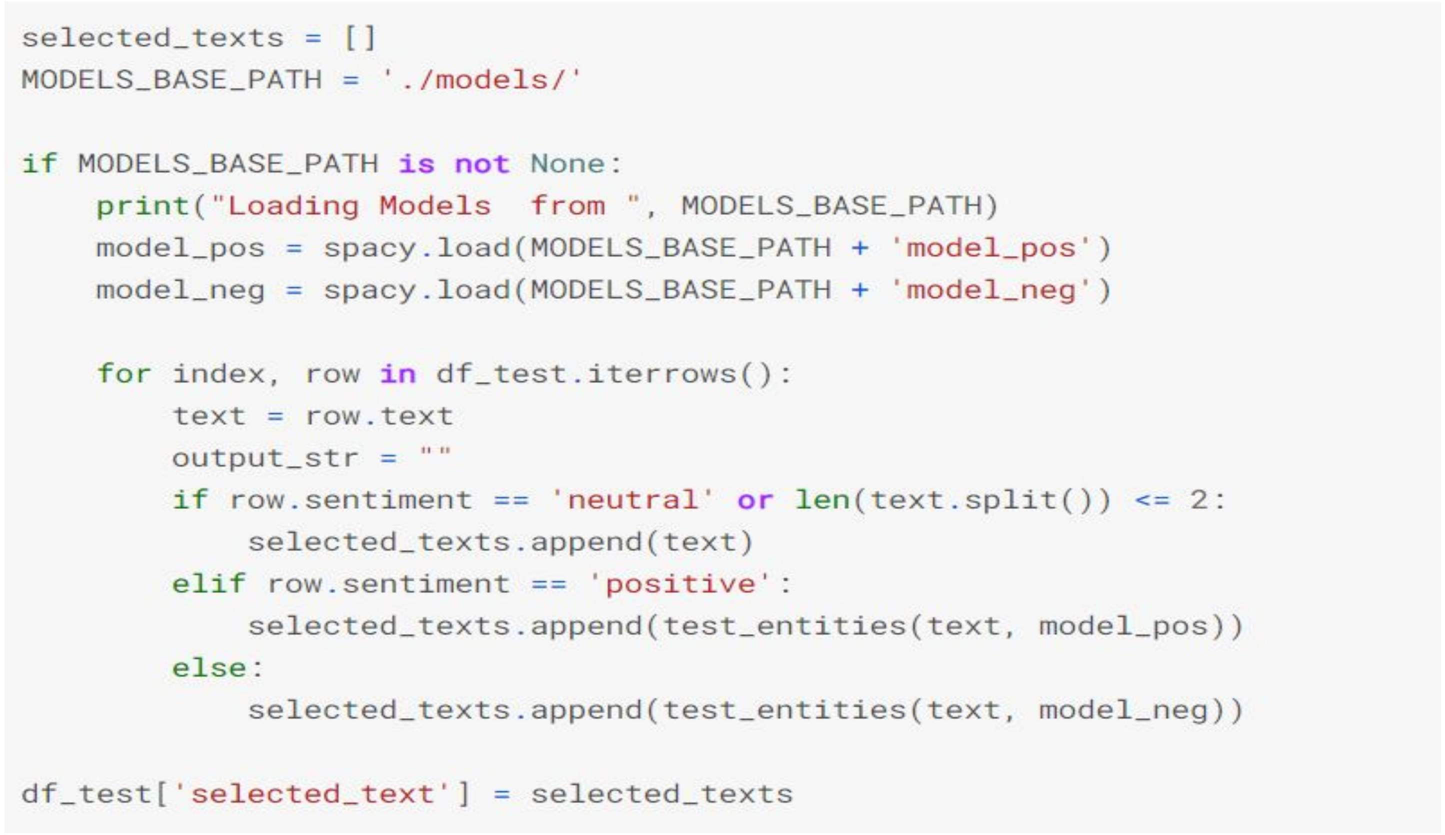


```
sentiment = 'positive'

train_data = get_training_data(sentiment)
model_path = get_model_out_path(sentiment)
train_model(train_data, model_path, n_iter=40)
```



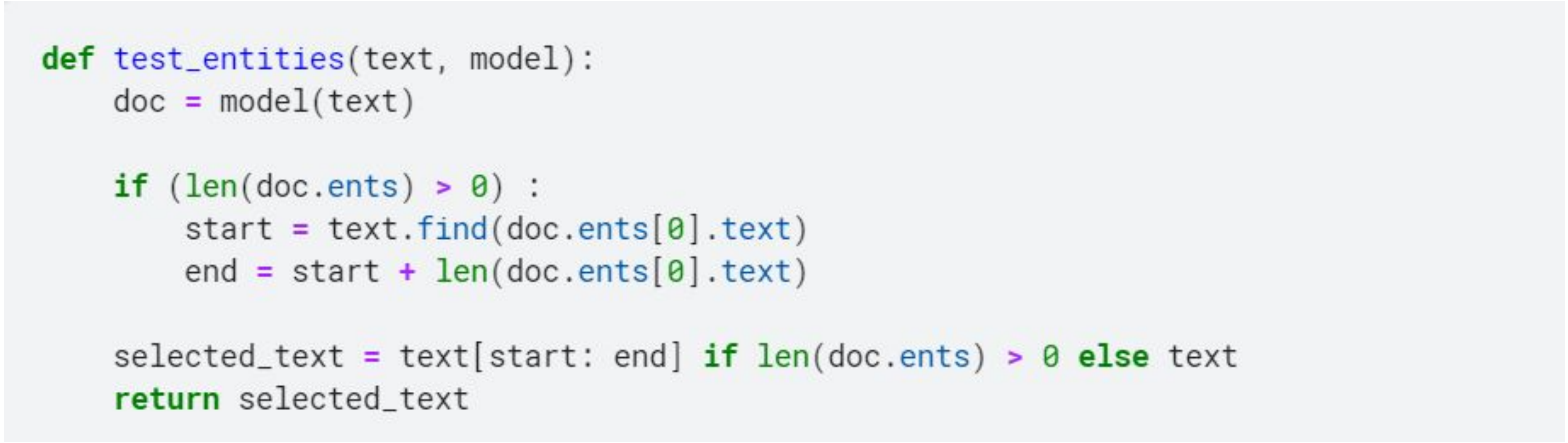
```
sentiment = 'negative'

train_data = get_training_data(sentiment)
model_path = get_model_out_path(sentiment)

train_model(train_data, model_path, n_iter=40)
```

Predict the selected text to the test tweets :



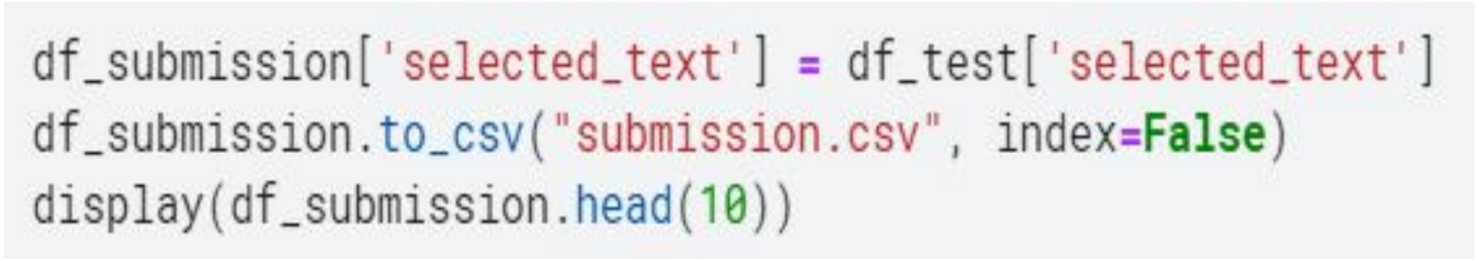
```python
selected_texts = []
MODELS_BASE_PATH = './models/'

if MODELS_BASE_PATH is not None:
    print("Loading Models  from ", MODELS_BASE_PATH)
    model_pos = spacy.load(MODELS_BASE_PATH + 'model_pos')
    model_neg = spacy.load(MODELS_BASE_PATH + 'model_neg')

    for index, row in df_test.iterrows():
        text = row.text
        output_str = ""
        if row.sentiment == 'neutral' or len(text.split()) <= 2:
            selected_texts.append(text)
        elif row.sentiment == 'positive':
            selected_texts.append(test_entities(text, model_pos))
        else:
            selected_texts.append(test_entities(text, model_neg))

df_test['selected_text'] = selected_texts
```

Predict the selected text :



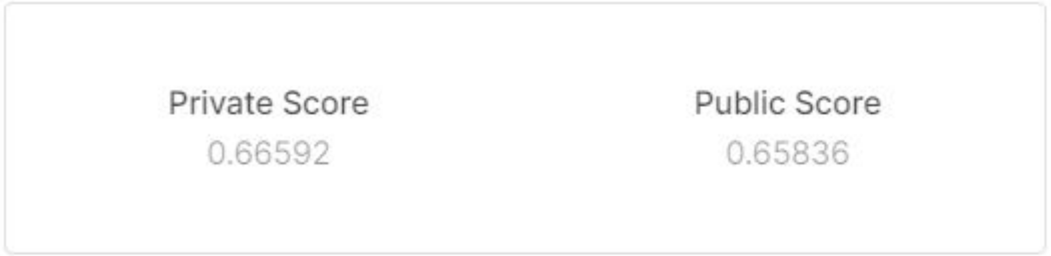
```python
def test_entities(text, model):
    doc = model(text)

    if (len(doc.ents) > 0) :
        start = text.find(doc.ents[0].text)
        end = start + len(doc.ents[0].text)

    selected_text = text[start: end] if len(doc.ents) > 0 else text
    return selected_text
```

Submit the model :



```
df_submission['selected_text'] = df_test['selected_text']
df_submission.to_csv("submission.csv", index=False)
display(df_submission.head(10))
```

Submission result :

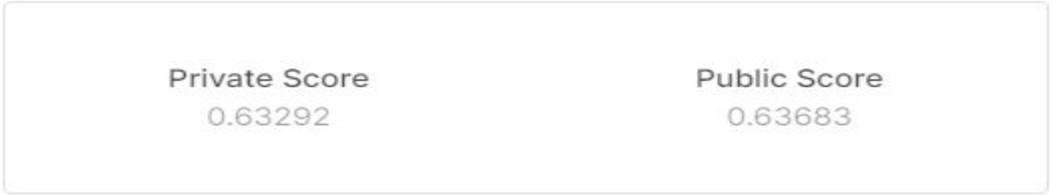


Private Score
0.66592

Public Score
0.65836

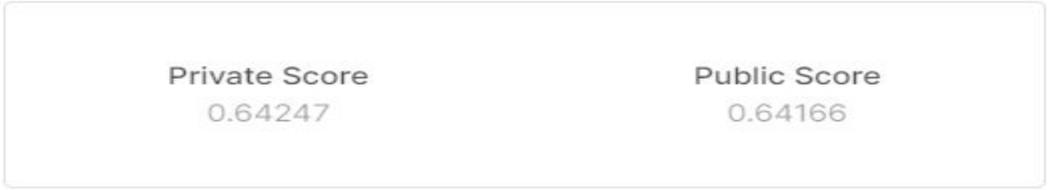
Changing number of iterations of train model :

n_iter = 1


Private Score 0.63292

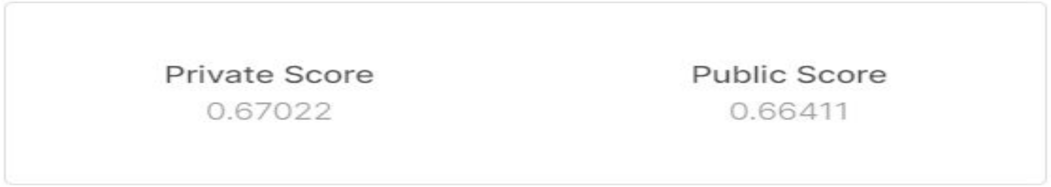
Public Score 0.63683

n_iter = 5


Private Score 0.64247

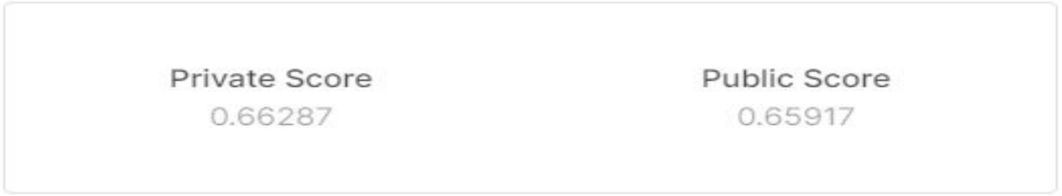
Public Score 0.64166

n_iter = 50


Private Score 0.67022

Public Score 0.66411

n_iter = 60


Private Score 0.66287

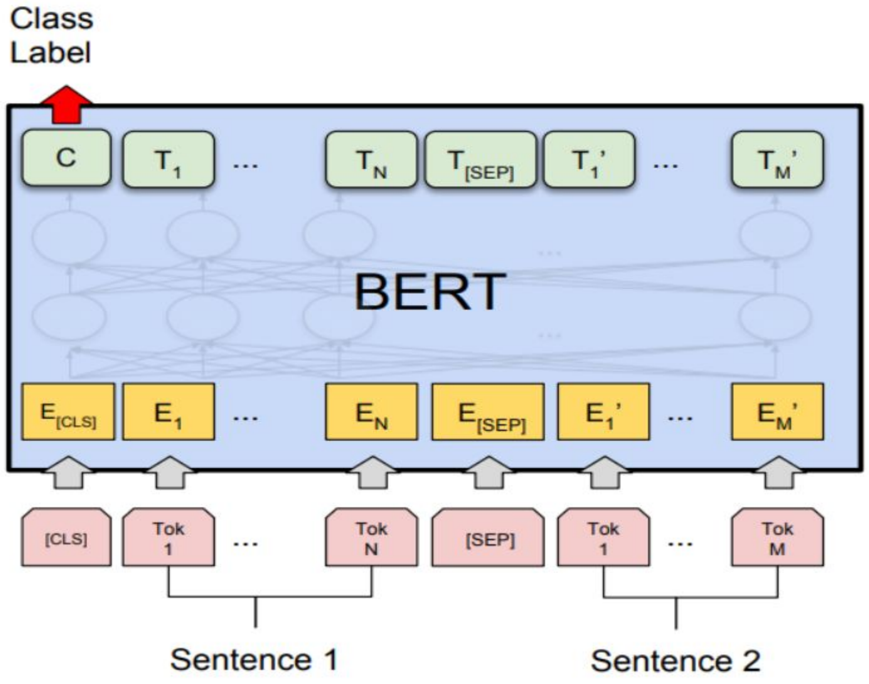
Public Score 0.65917

Only in range of 60 % :

Let’s change the model

BERT Language Model :

- BERT “**Bidirectional Encoder Representations from Transformers**” is an open source machine learning framework for NLP.
- It is based on **Transformers**, a deep learning model in which every output element is connected to every input element, and the weightings between them are **dynamically** calculated based upon their connection.
- Historically, language models could only read text input sequentially either left-to-right or right-to-left but couldn't do both at the same time. BERT is **different** because it is designed to read in **both** directions at once.



RoBERTa Language Model :

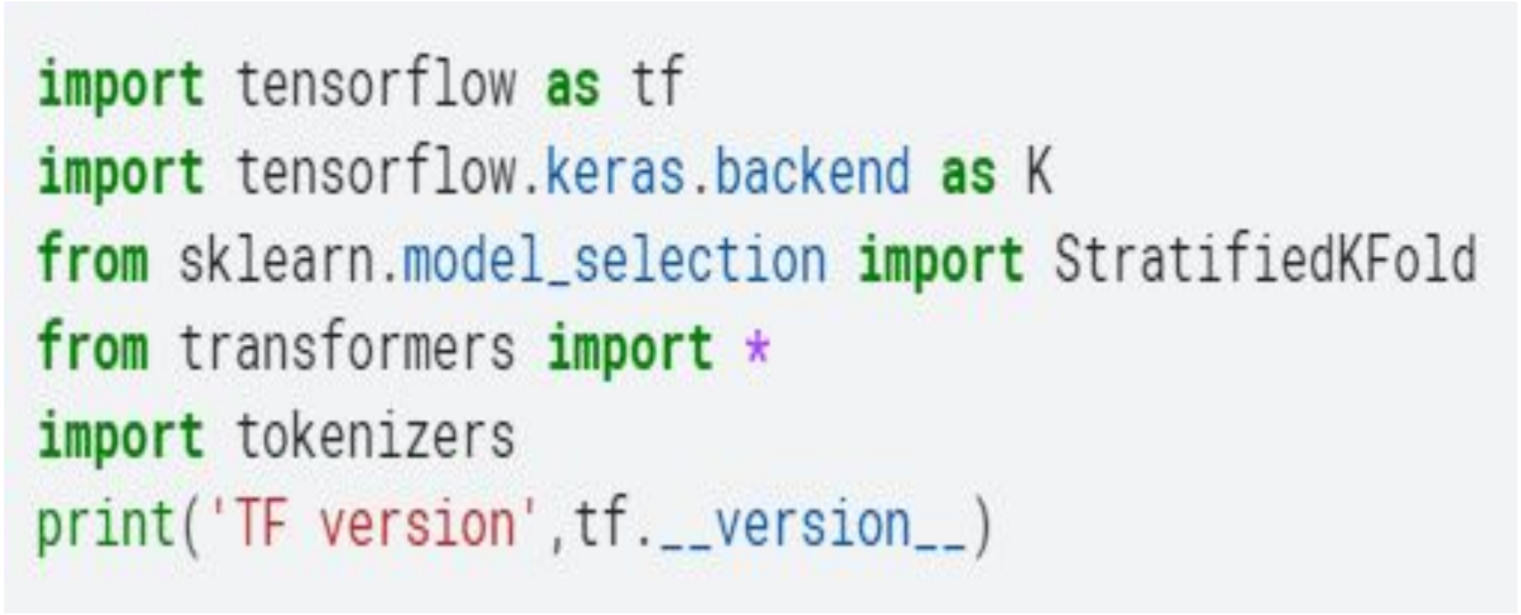
- RoBERTa “Robustly Optimized BERT Pretraining Approach” .
- It builds on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates.
- RoBERTa has the same architecture as BERT, but uses a byte-level BPE as a tokenizer and uses a different pre training scheme.
- The goal of RoBERTa is to optimize the training of BERT architecture in order to take lesser time during pre-training .

To call Roberta model :

Data should be prepared as follow :

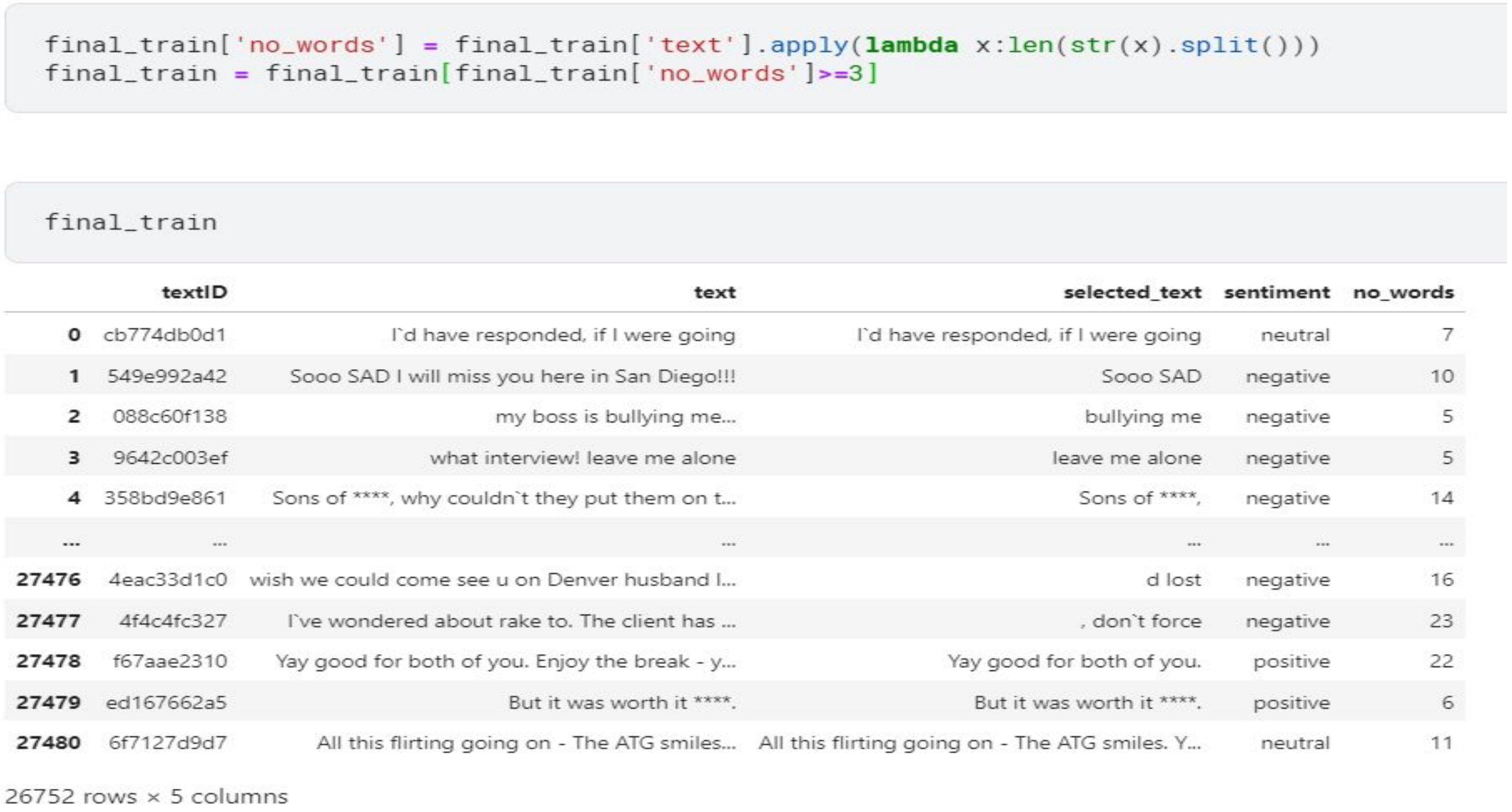
- Input_ids : [0] + id of text + [2,2] + id of selected text + [2]
- Attention mask :only length of Input_ids is set to 1 as The loss function while calculating it considers only the prediction of masked values and ignores the prediction of the non-masked values .
- Start_tokens : vector of zeros except for the index of start of selected text in the text plus 1 .
- End_tokens : vector of zeros except for the index of end of selected text in the text plus 1 .

Import another libraries :



```
import tensorflow as tf
import tensorflow.keras.backend as K
from sklearn.model_selection import StratifiedKFold
from transformers import *
import tokenizers
print('TF version',tf.__version__)
```

Same first steps as the previous model :



```python
final_train['no_words'] = final_train['text'].apply(lambda x:len(str(x).split()))
final_train = final_train[final_train['no_words']>=3]
```

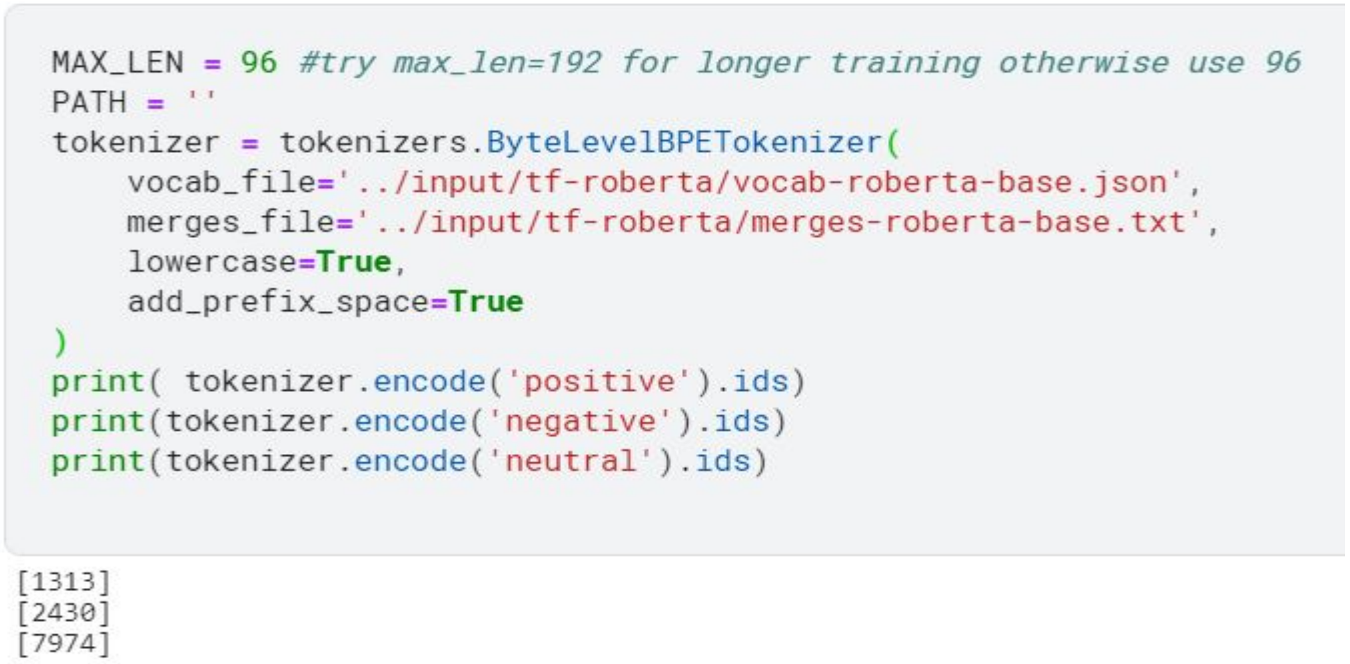
```python
final_train
```

	textID	text	selected_text	sentiment	no_words
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral	7
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative	10
2	088c60f138	my boss is bullying me...	bullying me	negative	5
3	9642c003ef	what interview! leave me alone	leave me alone	negative	5
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative	14
...	...	...	...	...	...
27476	4eac33d1c0	wish we could come see u on Denver husband l...	d lost	negative	16
27477	4f4c4fc327	I`ve wondered about rake to. The client has ...	, don`t force	negative	23
27478	f67aae2310	Yay good for both of you. Enjoy the break - y...	Yay good for both of you.	positive	22
27479	ed167662a5	But it was worth it ****.	But it was worth it ****.	positive	6
27480	6f7127d9d7	All this flirting going on - The ATG smiles...	All this flirting going on - The ATG smiles. Y...	neutral	11

26752 rows × 5 columns

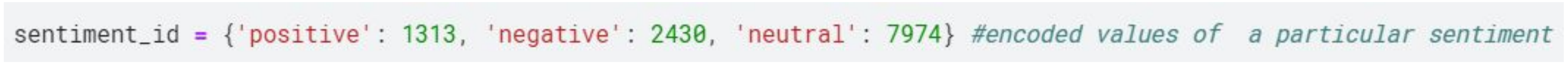
Set the settings of the tokenizer using Roberta files :

- Give an id for each word and here save the ids of (pos/neg/neu) words



```
MAX_LEN = 96 #try max_len=192 for longer training otherwise use 96
PATH = ''
tokenizer = tokenizers.ByteLevelBPETokenizer(
    vocab_file='../input/tf-roberta/vocab-roberta-base.json',
    merges_file='../input/tf-roberta/merges-roberta-base.txt',
    lowercase=True,
    add_prefix_space=True
)
print( tokenizer.encode('positive').ids)
print(tokenizer.encode('negative').ids)
print(tokenizer.encode('neutral').ids)
```

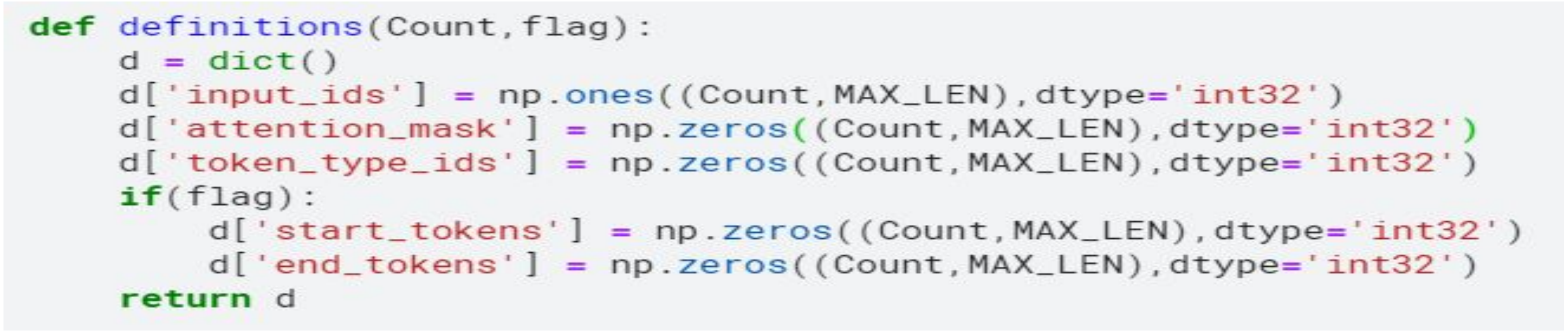
```
[1313]
[2430]
[7974]
```



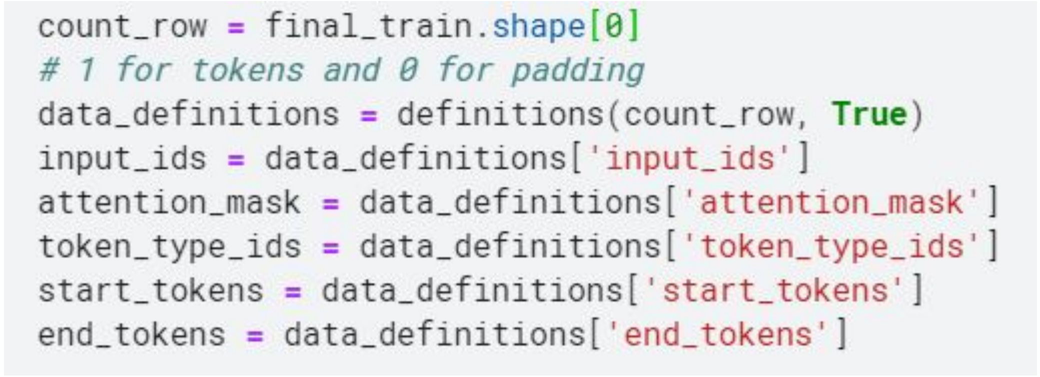
```
sentiment_id = {'positive': 1313, 'negative': 2430, 'neutral': 7974} #encoded values of  a particular sentiment
```

Initializing needed data :

- Initialize the start_tokens and the end_tokens only if there exists selected text “flag is true “ .

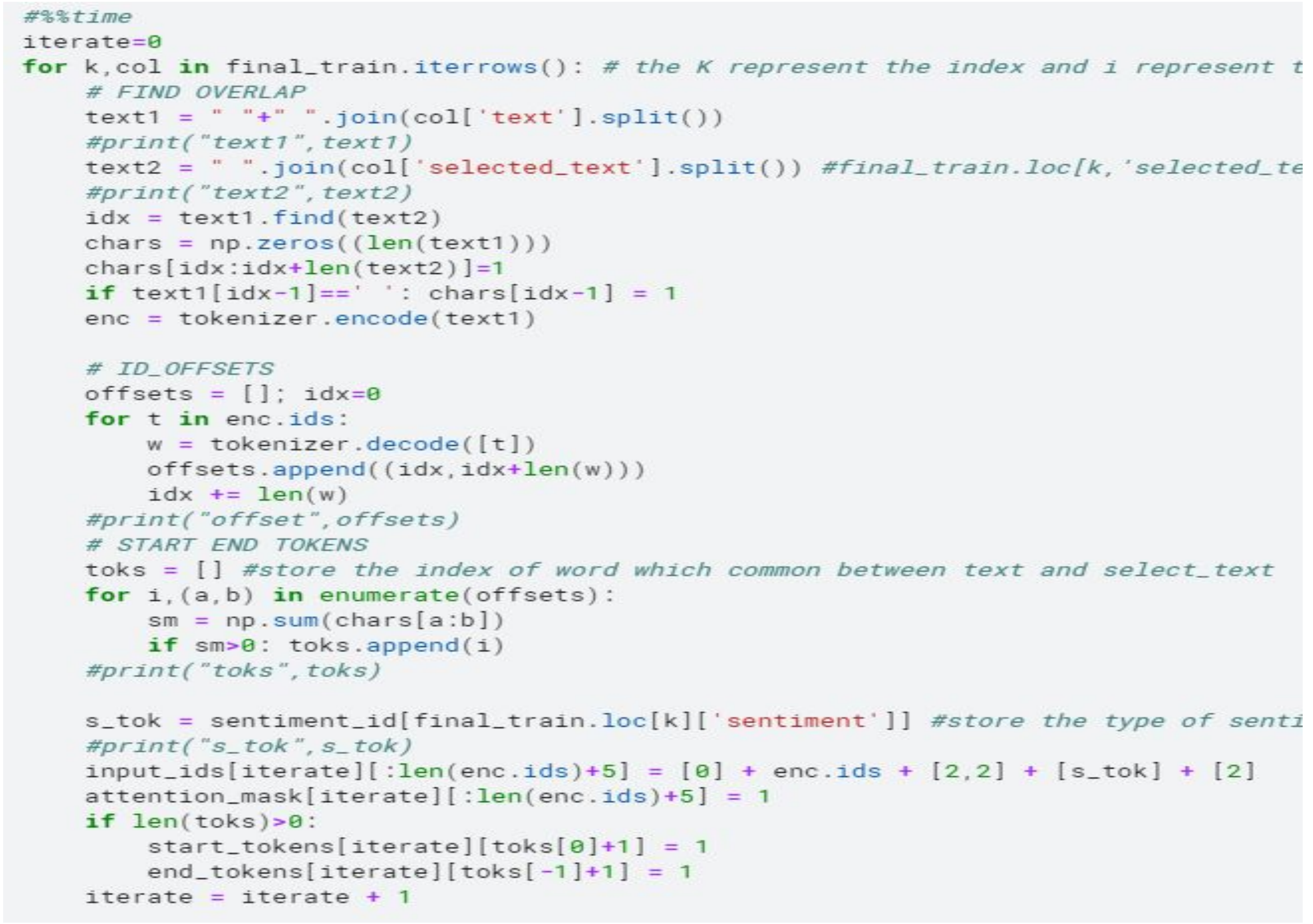


```
def definitions(Count,flag):
    d = dict()
    d['input_ids'] = np.ones((Count,MAX_LEN),dtype='int32')
    d['attention_mask'] = np.zeros((Count,MAX_LEN),dtype='int32')
    d['token_type_ids'] = np.zeros((Count,MAX_LEN),dtype='int32')
    if(flag):
        d['start_tokens'] = np.zeros((Count,MAX_LEN),dtype='int32')
        d['end_tokens'] = np.zeros((Count,MAX_LEN),dtype='int32')
    return d
```



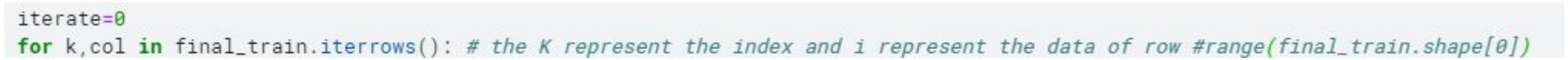
```
count_row = final_train.shape[0]
# 1 for tokens and 0 for padding
data_definitions = definitions(count_row, True)
input_ids = data_definitions['input_ids']
attention_mask = data_definitions['attention_mask']
token_type_ids = data_definitions['token_type_ids']
start_tokens = data_definitions['start_tokens']
end_tokens = data_definitions['end_tokens']
```

Preparing Needed data :



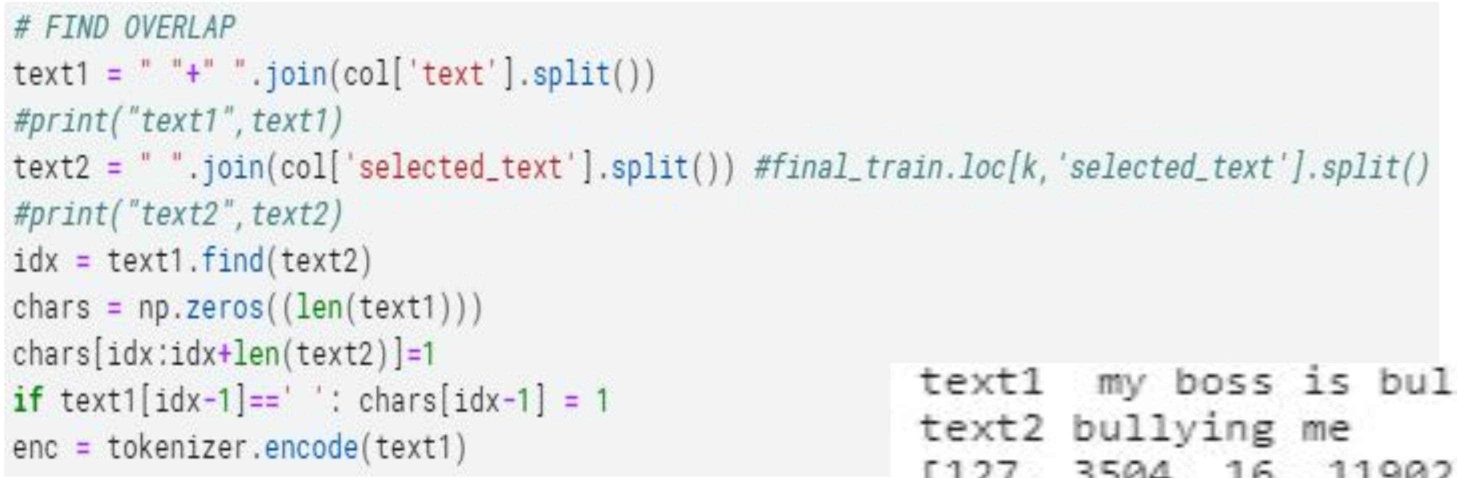
```
#%%time
iterate=0
for k,col in final_train.iterrows(): # the K represent the index and i represent t
    # FIND OVERLAP
    text1 = " "+" ".join(col['text'].split())
    #print("text1",text1)
    text2 = " ".join(col['selected_text'].split()) #final_train.loc[k,'selected_te
    #print("text2",text2)
    idx = text1.find(text2)
    chars = np.zeros((len(text1)))
    chars[idx:idx+len(text2)]=1
    if text1[idx-1]==' ': chars[idx-1] = 1
    enc = tokenizer.encode(text1)

    # ID_OFFSETS
    offsets = []; idx=0
    for t in enc.ids:
        w = tokenizer.decode([t])
        offsets.append((idx,idx+len(w)))
        idx += len(w)
    #print("offset",offsets)
    # START END TOKENS
    toks = [] #store the index of word which common between text and select_text
    for i,(a,b) in enumerate(offsets):
        sm = np.sum(chars[a:b])
        if sm>0: toks.append(i)
    #print("toks",toks)

    s_tok = sentiment_id[final_train.loc[k]['sentiment']] #store the type of senti
    #print("s_tok",s_tok)
    input_ids[iterate][:len(enc.ids)+5] = [0] + enc.ids + [2,2] + [s_tok] + [2]
    attention_mask[iterate][:len(enc.ids)+5] = 1
    if len(toks)>0:
        start_tokens[iterate][toks[0]+1] = 1
        end_tokens[iterate][toks[-1]+1] = 1
    iterate = iterate + 1
```

- **Looping in all rows of train data**

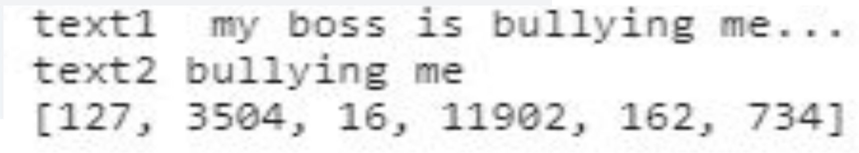


```
iterate=0
for k,col in final_train.iterrows(): # the K represent the index and i represent the data of row #range(final_train.shape[0])
```

- **Encode the text words and initialize chars array of length equals the text length with all zeros except the indices of selected text equals 1**

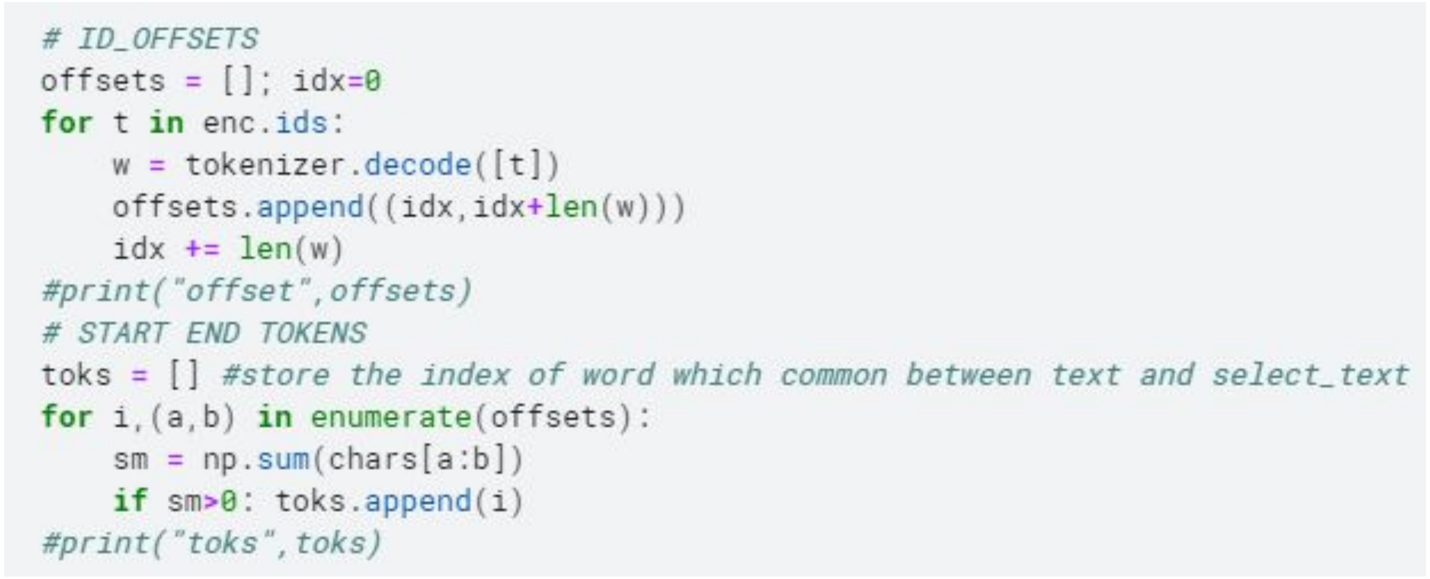


```
# FIND OVERLAP
text1 = " "+" ".join(col['text'].split())
#print("text1",text1)
text2 = " ".join(col['selected_text'].split()) #final_train.loc[k,'selected_text'].split()
#print("text2",text2)
idx = text1.find(text2)
chars = np.zeros((len(text1)))
chars[idx:idx+len(text2)]=1
if text1[idx-1]==' ': chars[idx-1] = 1
enc = tokenizer.encode(text1)
```

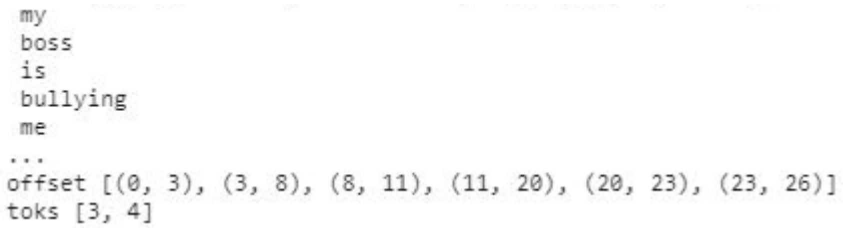


```
text1  my boss is bullying me...
text2 bullying me
[127, 3504, 16, 11902, 162, 734]
```

● Decode the text , store the offsets and store indices of selected in toks

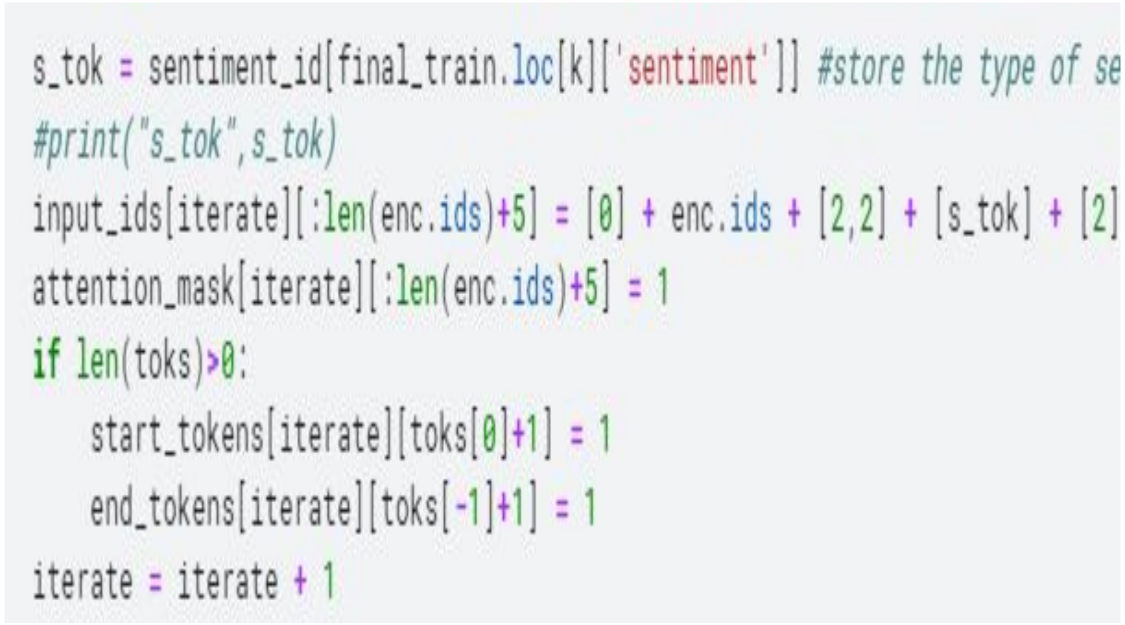


```
# ID_OFFSETS
offsets = []; idx=0
for t in enc.ids:
    w = tokenizer.decode([t])
    offsets.append((idx,idx+len(w)))
    idx += len(w)
#print("offset",offsets)
# START END TOKENS
toks = [] #store the index of word which common between text and select_text
for i,(a,b) in enumerate(offsets):
    sm = np.sum(chars[a:b])
    if sm>0: toks.append(i)
#print("toks",toks)
```

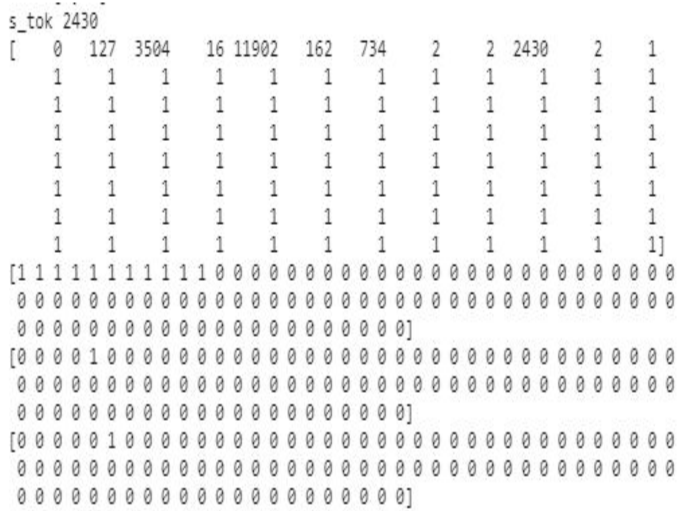


```
  my
  boss
  is
  bullying
  me
...
offset [(0, 3), (3, 8), (8, 11), (11, 20), (20, 23), (23, 26)]
toks [3, 4]
```

- **Set sentiment id of every train row in s_tok , use it to combine the input_ids , set the attention mask / start_tokens /end_tokens and add one to iterate**

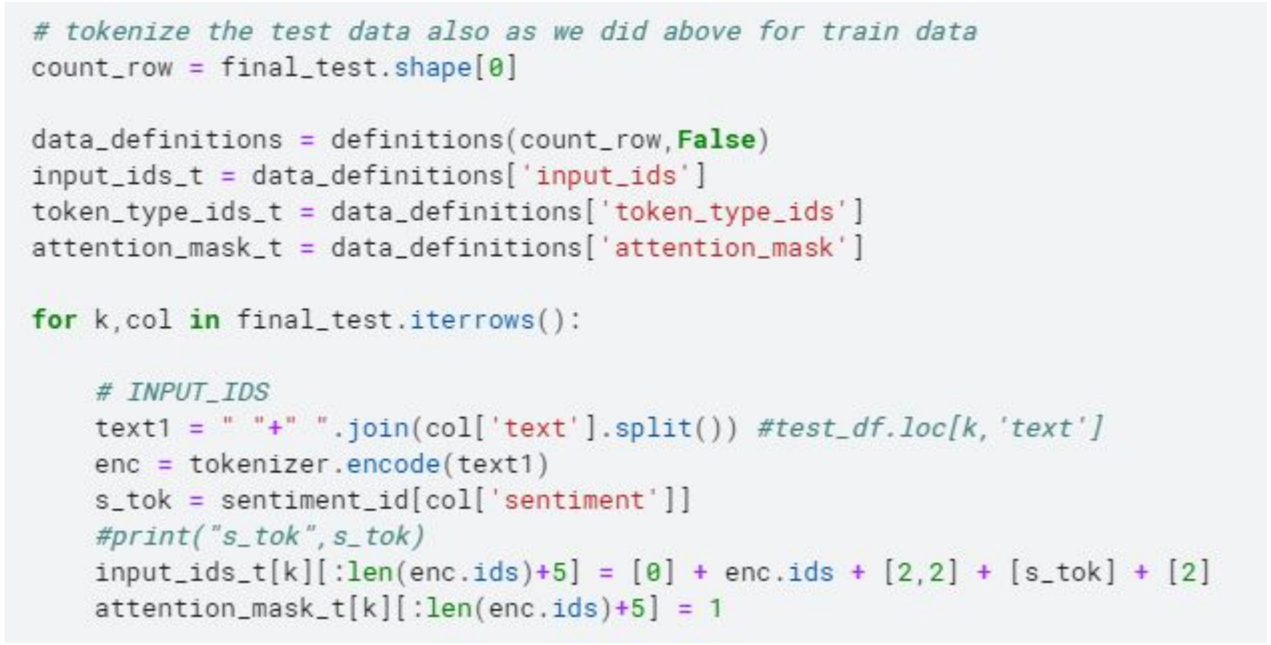


```
s_tok = sentiment_id[final_train.loc[k]['sentiment']] #store the type of se
#print("s_tok",s_tok)
input_ids[iterate][:len(enc.ids)+5] = [0] + enc.ids + [2,2] + [s_tok] + [2]
attention_mask[iterate][:len(enc.ids)+5] = 1
if len(toks)>0:
    start_tokens[iterate][toks[0]+1] = 1
    end_tokens[iterate][toks[-1]+1] = 1
iterate = iterate + 1
```



```
s_tok 2430
[    0   127  3504    16 11902   162   734     2     2  2430     2     1
     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1]
[1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

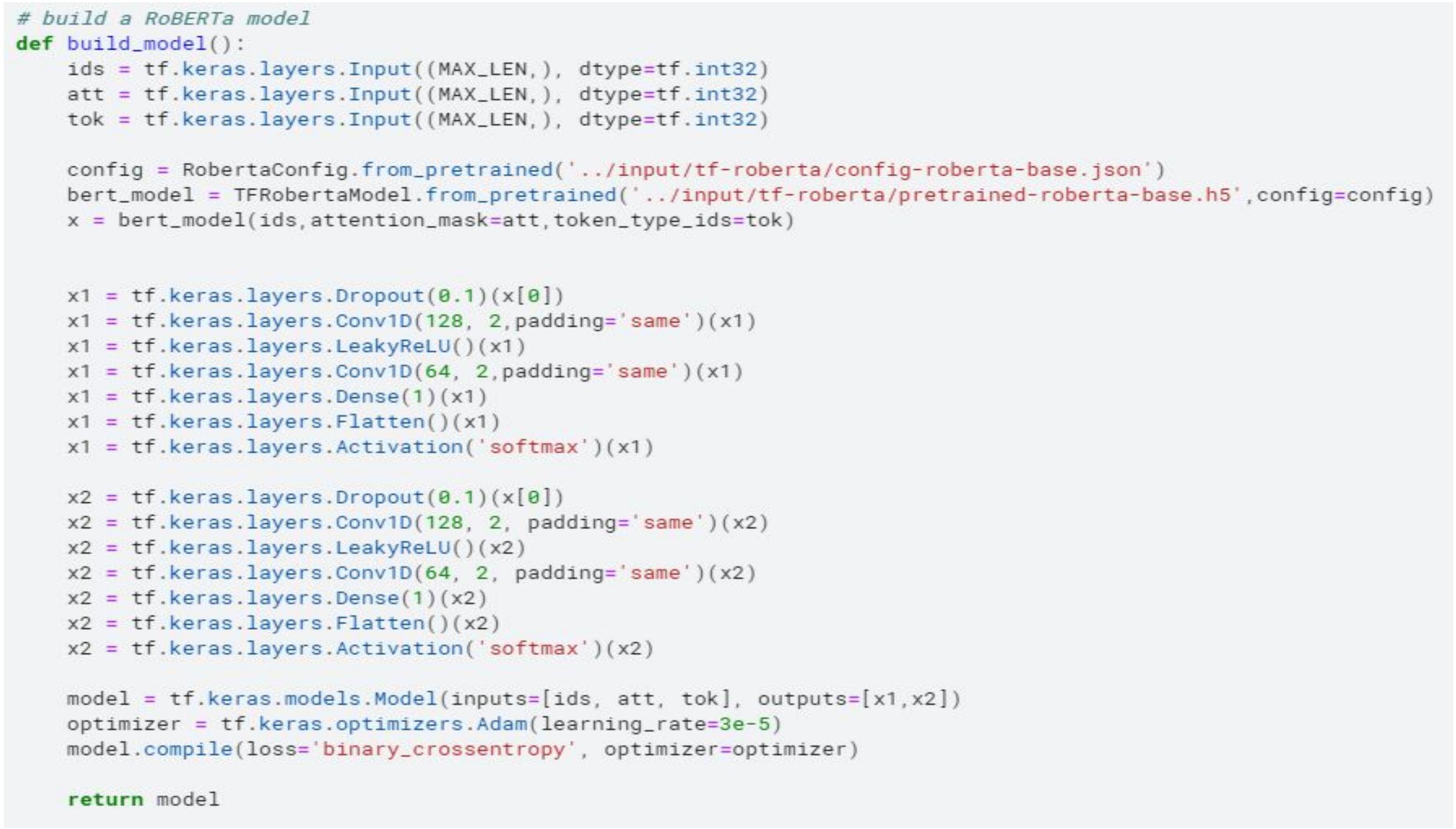
Do the same but for test data



```python
# tokenize the test data also as we did above for train data
count_row = final_test.shape[0]

data_definitions = definitions(count_row,False)
input_ids_t = data_definitions['input_ids']
token_type_ids_t = data_definitions['token_type_ids']
attention_mask_t = data_definitions['attention_mask']

for k,col in final_test.iterrows():

    # INPUT_IDS
    text1 = " "+" ".join(col['text'].split()) #test_df.loc[k,'text']
    enc = tokenizer.encode(text1)
    s_tok = sentiment_id[col['sentiment']]
    #print("s_tok",s_tok)
    input_ids_t[k][:len(enc.ids)+5] = [0] + enc.ids + [2,2] + [s_tok] + [2]
    attention_mask_t[k][:len(enc.ids)+5] = 1
```

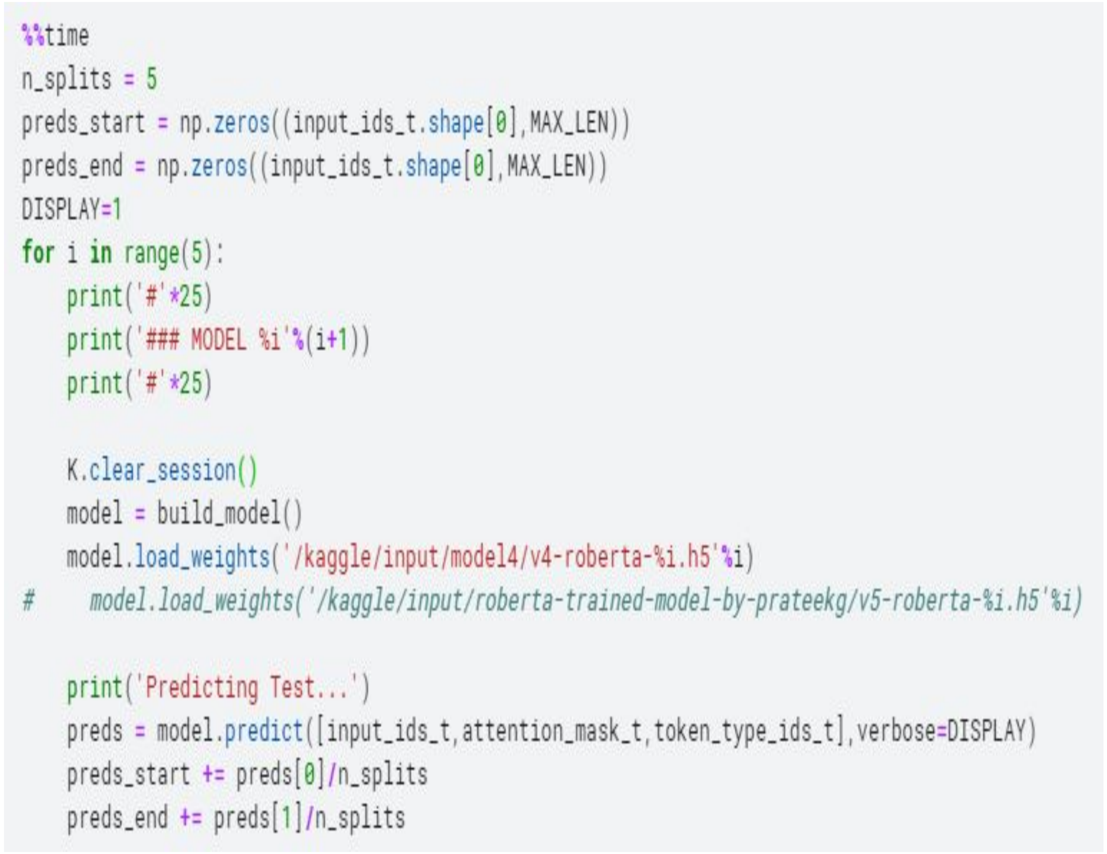
Note : no selected text so there doesn’t exist start_tokens and end_tokens .

● The model :

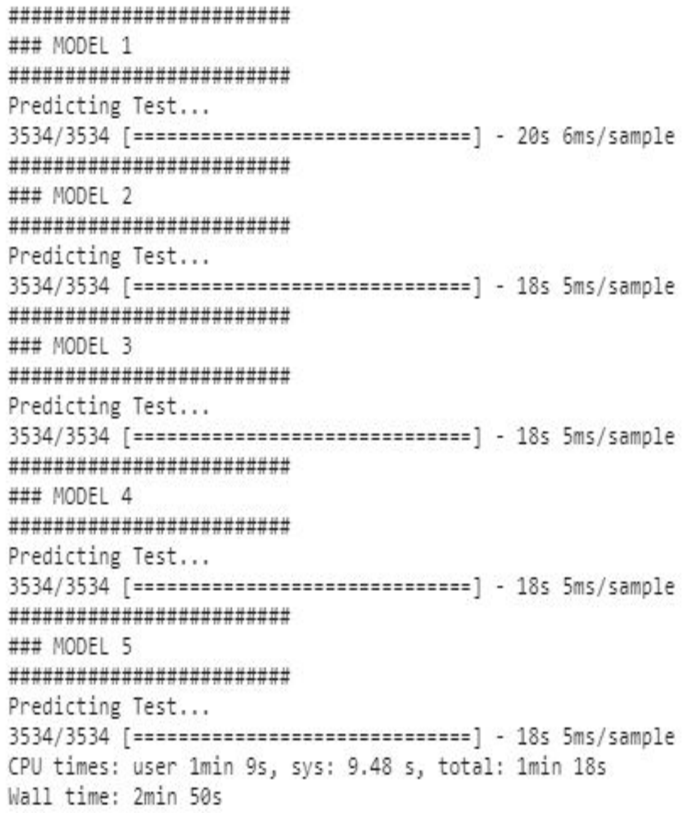


```python
# build a RoBERTa model
def build_model():
    ids = tf.keras.layers.Input((MAX_LEN,), dtype=tf.int32)
    att = tf.keras.layers.Input((MAX_LEN,), dtype=tf.int32)
    tok = tf.keras.layers.Input((MAX_LEN,), dtype=tf.int32)

    config = RobertaConfig.from_pretrained('../input/tf-roberta/config-roberta-base.json')
    bert_model = TFRobertaModel.from_pretrained('../input/tf-roberta/pretrained-roberta-base.h5',config=config)
    x = bert_model(ids,attention_mask=att,token_type_ids=tok)


    x1 = tf.keras.layers.Dropout(0.1)(x[0])
    x1 = tf.keras.layers.Conv1D(128, 2,padding='same')(x1)
    x1 = tf.keras.layers.LeakyReLU()(x1)
    x1 = tf.keras.layers.Conv1D(64, 2,padding='same')(x1)
    x1 = tf.keras.layers.Dense(1)(x1)
    x1 = tf.keras.layers.Flatten()(x1)
    x1 = tf.keras.layers.Activation('softmax')(x1)

    x2 = tf.keras.layers.Dropout(0.1)(x[0])
    x2 = tf.keras.layers.Conv1D(128, 2, padding='same')(x2)
    x2 = tf.keras.layers.LeakyReLU()(x2)
    x2 = tf.keras.layers.Conv1D(64, 2, padding='same')(x2)
    x2 = tf.keras.layers.Dense(1)(x2)
    x2 = tf.keras.layers.Flatten()(x2)
    x2 = tf.keras.layers.Activation('softmax')(x2)

    model = tf.keras.models.Model(inputs=[ids, att, tok], outputs=[x1,x2])
    optimizer = tf.keras.optimizers.Adam(learning_rate=3e-5)
    model.compile(loss='binary_crossentropy', optimizer=optimizer)

    return model
```

● We built the model for 5 times :

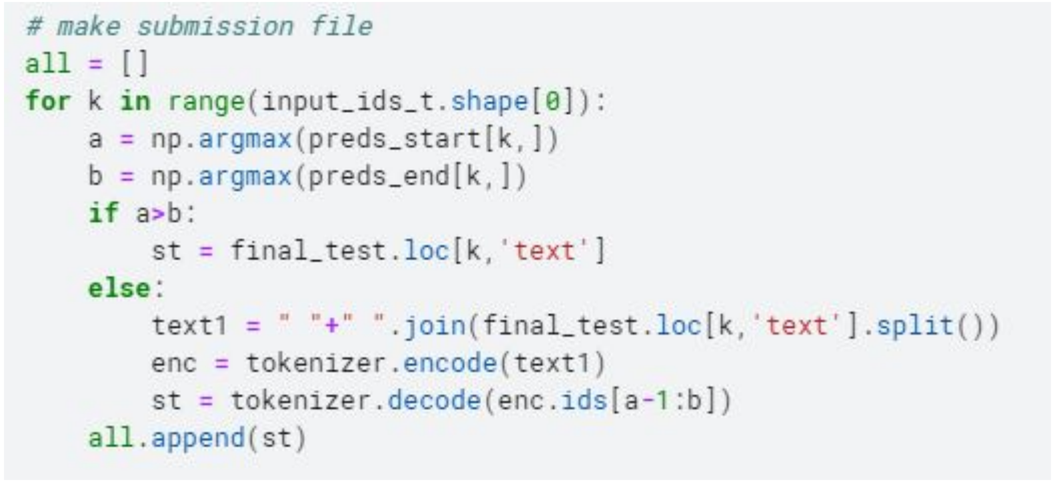


```
%%time
n_splits = 5
preds_start = np.zeros((input_ids_t.shape[0],MAX_LEN))
preds_end = np.zeros((input_ids_t.shape[0],MAX_LEN))
DISPLAY=1
for i in range(5):
    print('#'*25)
    print('### MODEL %i'%(i+1))
    print('#'*25)

    K.clear_session()
    model = build_model()
    model.load_weights('/kaggle/input/model4/v4-roberta-%i.h5'%i)
#   model.load_weights('/kaggle/input/roberta-trained-model-by-prateekg/v5-roberta-%i.h5'%i)

    print('Predicting Test...')
    preds = model.predict([input_ids_t,attention_mask_t,token_type_ids_t],verbose=DISPLAY)
    preds_start += preds[0]/n_splits
    preds_end += preds[1]/n_splits
```

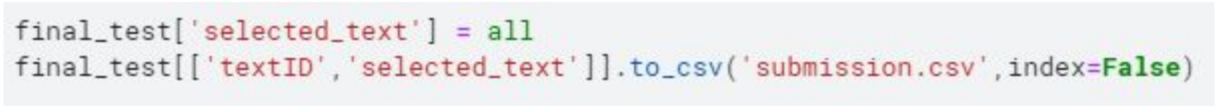


```
#########################
### MODEL 1
#########################
Predicting Test...
3534/3534 [==============================] - 20s 6ms/sample
#########################
### MODEL 2
#########################
Predicting Test...
3534/3534 [==============================] - 18s 5ms/sample
#########################
### MODEL 3
#########################
Predicting Test...
3534/3534 [==============================] - 18s 5ms/sample
#########################
### MODEL 4
#########################
Predicting Test...
3534/3534 [==============================] - 18s 5ms/sample
#########################
### MODEL 5
#########################
Predicting Test...
3534/3534 [==============================] - 18s 5ms/sample
CPU times: user 1min 9s, sys: 9.48 s, total: 1min 18s
Wall time: 2min 50s
```

● Make submission file ：

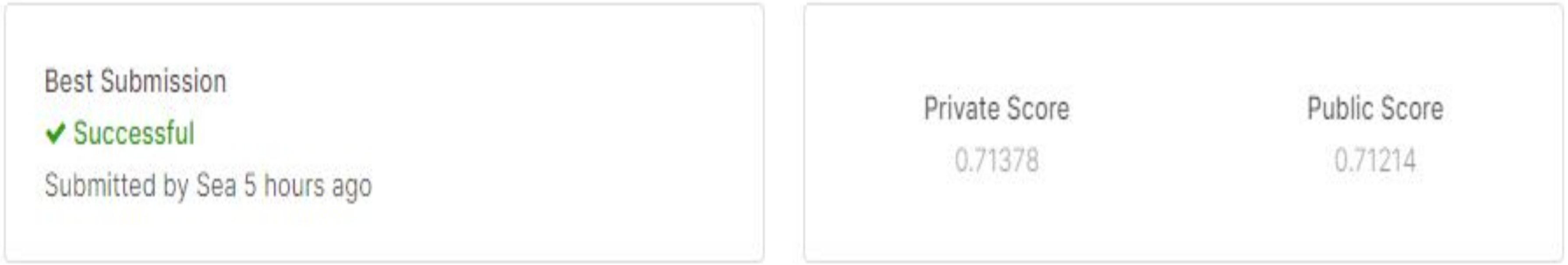


```
# make submission file
all = []
for k in range(input_ids_t.shape[0]):
    a = np.argmax(preds_start[k,])
    b = np.argmax(preds_end[k,])
    if a>b:
        st = final_test.loc[k,'text']
    else:
        text1 = " "+" ".join(final_test.loc[k,'text'].split())
        enc = tokenizer.encode(text1)
        st = tokenizer.decode(enc.ids[a-1:b])
    all.append(st)
```



```
final_test['selected_text'] = all
final_test[['textID','selected_text']].to_csv('submission.csv',index=False)
```

- **The Score of the submission :**



Best Submission

✓ Successful

Submitted by Sea 5 hours ago

Private Score	Public Score
0.71378	0.71214

Thanks