Final Project w266: Toxicity Detection

> Jake Pistotnik & Vijay Rangantha

Installs

```
#Installs
!pip install pydot --quiet
!pip install gensim==3.8.3 --quiet
!pip install tensorflow-datasets --quiet
!pip install -U tensorflow-text==2.8.2 --quiet
!pip install transformers --quiet
!pip install datasets
!pip3 install emoji==0.6.0
```

```
Requirement already satisfied: packaging in /usr/local/lib/python3.9/dist-packag
Collecting responses<0.19
  Downloading responses-0.18.0-py3-none-any.whl (38 kB)
Requirement already satisfied: huggingface-hub<1.0.0,>=0.11.0 in /usr/local/lib/
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.9/dist-packa
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: fsspec[http]>=2021.11.1 in /usr/local/lib/python3
Collecting xxhash
  Downloading xxhash-3.2.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.w
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.2/212.2 kB 21.3 MB/s eta 0:00:00
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.9/dist-pa
Collecting aiohttp
  Downloading aiohttp-3.8.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64
```

```
Collecting aiosignal>=1.1.2
  Downloading aiosignal-1.3.1-py3-none-any.whl (7.6 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/pyth
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/di
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-pac
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages
Installing collected packages: xxhash, multidict, frozenlist, dill, async-timeou
Successfully installed aiohttp-3.8.4 aiosignal-1.3.1 async-timeout-4.0.2 dataset
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Collecting emoji==0.6.0
  Downloading emoji-0.6.0.tar.gz (51 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 51.0/51.0 kB 3.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: emoji
  Building wheel for emoji (setup.py) ... done
  Created wheel for emoji: filename=emoji-0.6.0-py3-none-any.whl size=49732 sha2
  Stored in directory: /root/.cache/pip/wheels/70/2a/7f/1a0012c86b1061c6ee2ed956
Successfully built emoji
Installing collected packages: emoji
Successfully installed emoji-0.6.0
```

Imports

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

from tensorflow.keras.layers import Embedding, Input, Dense, Lambda
from tensorflow.keras.models import Model
import tensorflow.keras.backend as K
import tensorflow_datasets as tfds
import tensorflow_text as tf_text

from transformers import BertTokenizer, TFBertModel, AutoModel, AutoTokenizer, TFAuto

import nltk
from nltk.corpus import reuters
from nltk.data import find
from nltk.tokenize import RegexpTokenizer
from nltk.tokenize import wordpunct_tokenize
from nltk.tokenize import TweetTokenizer

import sklearn as sk
import os

import matplotlib.pyplot as plt
```

```
import re

#This continues to work with gensim 3.8.3.  It doesn't yet work with 4.x.
#Make sure your pip install command specifies gensim==3.8.3
import gensim

from sklearn.metrics import classification_report
import seaborn as sns
```

DataSet Import

```
import datasets
dataset = datasets.load_dataset('ucberkeley-dlab/measuring-hate-speech', 'binary')
df = dataset['train'].to_pandas()

##Citation
##@article{kennedy2020constructing,
  #title={Constructing interval variables via faceted Rasch measurement and multitask
  #author={Kennedy, Chris J and Bacon, Geoff and Sahn, Alexander and von Vacano, Claud
  #journal={arXiv preprint arXiv:2009.10277},
  #year={2020}
#}
```

```
Downloading readme: 100% 4.03k/4.03k [00:00<00:00, 180kB/s]
Downloading and preparing dataset parquet/ucberkeley-dlab--measuring-hate-speech
Downloading data files: 100% 1/1 [00:00<00:00, 1.47it/s]
Downloading data: 100% 14.1M/14.1M [00:00<00:00, 44.3MB/s]
Extracting data files: 100% 1/1 [00:00<00:00, 47.67it/s]
Dataset parquet downloaded and prepared to /root/.cache/huggingface/datasets/ucb
```

```
# list of all the columns in the dataframe
for i in df.columns:
  print(i)
```

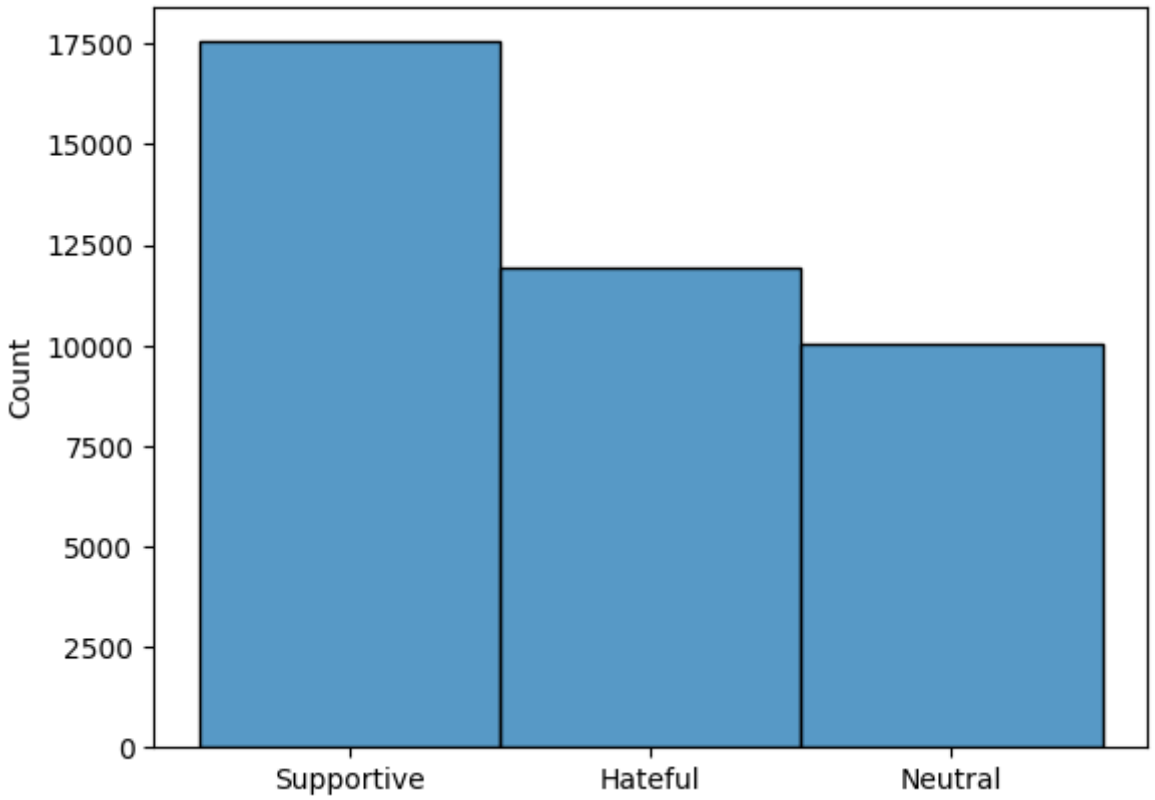
```
    annotator_ideology
    annotator_gender_men
    annotator_gender_women
    annotator_gender_non_binary
    annotator_gender_prefer_not_to_say
    annotator_gender_self_describe
    annotator_transgender
    annotator_cisgender
    annotator_transgender_prefer_not_to_say
    annotator_education_some_high_school
    annotator_education_high_school_grad
    annotator_education_some_college
    annotator_education_college_grad_aa
    annotator_education_college_grad_ba
    annotator_education_professional_degree
    annotator_education_masters
    annotator_education_phd
    annotator_income_<10k
    annotator_income_10k-50k
    annotator_income_50k-100k
    annotator_income_100k-200k
    annotator_income_>200k
    annotator_ideology_extremeley_conservative
    annotator_ideology_conservative
    annotator_ideology_slightly_conservative
    annotator_ideology_neutral
    annotator_ideology_slightly_liberal
    annotator_ideology_liberal
    annotator_ideology_extremeley_liberal
    annotator_ideology_no_opinion
    annotator_race_asian
    annotator_race_black
    annotator_race_latinx
    annotator_race_middle_eastern
    annotator_race_native_american
    annotator_race_pacific_islander
    annotator_race_white
    annotator_race_other
    annotator_age
    annotator_religion_atheist
    annotator_religion_buddhist
    annotator_religion_christian
    annotator_religion_hindu
    annotator_religion_jewish
    annotator_religion_mormon
    annotator_religion_muslim
    annotator_religion_nothing
    annotator_religion_other
    annotator_sexuality_bisexual
    annotator_sexuality_gay
    annotator_sexuality_straight
    annotator_sexuality_other

# creating arrays of text column and labels column for viewing and making text columns
df['text'].astype(str)
df = df[['text', 'hate_speech_score']].groupby('text').mean().reset_index()
```

```
text = np.array(df['text'])
labels = np.array(df['hate_speech_score'])
```

EDA

```
##EDA
import seaborn as sns
def str_labler(arr):
  new_arr = []
  for i in arr:
    if i >= .3:
      new_arr.append("Hateful")
    elif i < .3 and i > -1:
      new_arr.append("Neutral")
    else:
      new_arr.append("Supportive")
  return new_arr
labels_str = str_labler(labels)
sns.histplot(labels_str)
```

```
<Axes: ylabel='Count'>
```




```
supp_count = len([i for i in labels_str if i =="Supportive"])
hate_count = len([i for i in labels_str if i =="Hateful"])
neut_count = len([i for i in labels_str if i =="Neutral"])
total_count = len(labels_str)
print("Supportive Count: ", supp_count)
```

```
print("Supportive Prop: ", supp_count/total_count)
print("Hateful Count: ", hate_count)
print("Hateful Prop: ", hate_count/total_count)
print("Neutral Count: ", neut_count)
print("Neutral Prop: ", neut_count/total_count)
```

```
Supportive Count:  17553
Supportive Prop:  0.4436496903829142
Hateful Count:  11954
Hateful Prop:  0.3021357260204726
Neutral Count:  10058
Neutral Prop:  0.25421458359661314
```

```
supp_exs = df[df['hate_speech_score'] < -1].sort_values('hate_speech_score')
hate_exs = df[df['hate_speech_score'] > .3].sort_values('hate_speech_score', ascending
neut_exs = df[df['hate_speech_score'].between(-1,.3)]
hate_neut_edge_exs = df[df['hate_speech_score'].between(.3,.6)]
```

```
supp_exs
```

	text	hate_speech_score
11464	Fellow trans girl here! Love and support confi...	-8.34
29324	There's a church near where in live that used ...	-8.30
11687	For my brave girls all over the world defying ...	-8.28
24771	Pro wrestler Mercedes Martinez has been provin...	-8.24
27312	Such a huge and heartwarming thank you to over...	-8.21
...	...	...
22087	Minnesota...the Sweden of the United States. M...	-1.01
22995	Niggas listen to their boys too much. Girls do...	-1.01
38062	rape is horrible, rapists are scum. the proble...	-1.01
19972	It's not simply a label if it's true tho, you ...	-1.01
18216	IM GONNA GO HOME AND TWEET ABOUT THIS TO MY A*...	-1.01

17475 rows × 2 columns

```
hate_exs
```

	text	hate_speech_score
10421	Dig burn PITS and treat them THE INVADING NAZI...	6.30
36382	fr diva gays are the worst types of faggots. t...	6.09
20501	KILL THEM ALL PUT THEM HOES TO REST WALK IN BU...	6.05
16735	I wanna abuse that worthless whore ass for hours	6.01
23122	No fuck off. You are all lost causes and all o...	5.78
...	...	...
10907	Eat my pussy from the back bitch	0.31
18059	I'm sure your God is real proud there's a sexu...	0.31
11074	Europeans who are flooding their country with ...	0.31

```
neut_exs[neut_exs['hate_speech_score'].between(-.01,.01)]
```

	text	hate_speech_score
18	".. Soo what are you gonna pick love? ~" 💬 to ...	-0.01
229	"I'm not racist, ah jus think black people and...	0.01
297	"Men who are too good looking are never good i...	-0.01
527	"You'd die for her." There is the answer, you ...	-0.01
925	'Slaughter the Jews' painted on section of Wes...	0.00
...	...	...
38195	she should choke on my penis :)	0.01
38380	stop posting faggot	-0.01
38690	this bitch retarted URL	0.01
38899	ugly bitches talk the most shit like stfu	-0.01
39288	yeah, in America she would have been beaten to...	0.01

205 rows × 2 columns

```
hate_neut_edge_exs.sort_values('hate_speech_score', ascending = True).loc[23592]['text
```

```
'Number killed from car accidents: 1,000,000. Number killed by the word nigger:
0 Therefore, nigger is no big deal.'
```

Pre-Processing

```
def labler(arr):
  new_arr = []
  for i in arr:
    if i >= .3:
      new_arr.append(1)
    elif i < .3 and i > -1:
      new_arr.append(0)
    else:
      new_arr.append(2)
  return new_arr


#https://www.educative.io/answers/remove-all-the-punctuation-marks-from-a-sentence-usi
def preprocesser(ex):
  clean_data = []
  for i in ex:
    i = i.lower()
    j = re.sub(r'@[A-Za-z]*\.?[A-Za-z0-9]*', "", i)
    j = re.sub(r"http\S+", "", j)
    j = re.sub(r"url\S+", "", j)
    j = re.sub(r"!flair", "", j)
    j = re.sub(r"\[", "", j)
    j = re.sub(r"\]", "", j)
    while '  ' in j:
      j = j.replace('  ', ' ')
    clean_data.append(j)
  return np.array(clean_data)

df
```

	text	hate_speech_score
0	! thank u! im transmasc and generally present ...	-3.92
1	!Go fuck yourself faggot!	2.47
2	!flair [I love women and minorities]	-4.48
3	!flair [death to all niggers and gays]	3.28
4	" 'convoluted' genealogy of Jesus"; was that c...	-3.18
...	...	...
39560	🤧 The limousine liberal Jews and WASPs who wa...	1.99
39561	😡 CONGRATULATIONS #MARYLAND #BALTIMORE LIBERALS...	-0.51
39562	🥳 another bitch will when another bitch can't	0.51
39563	😉😉 next time stay your ass in the car before o...	-1.22
39564	🧬🧬🧬 *IM SMARTER THAN SSSNIPERWOLF.I KNOW EVERY...	-1.55

39565 rows × 2 columns

```
#labels = labler(labels)
new_df = df.apply(lambda x: labler(x) if x.name == 'hate_speech_score' else x,
                  axis = 0)
new_df = new_df.apply(lambda x: preprocesser(x) if x.name == 'text' else x,
                  axis = 0)
```

```
new_df['hate_speech_score'].value_counts()
```

```
2    17553
1    12730
0     9282
Name: hate_speech_score, dtype: int64
```

```
new_df['2class_hate_speech'] = [0 if i in [0,2] else 1 for i in new_df['hate_speech_sc
```

```
new_df['2class_hate_speech'].value_counts()
```

```
0    26835
1    12730
Name: 2class_hate_speech, dtype: int64
```

```
np.random.seed(6543)
grouped = new_df[['hate_speech_score', 'text', '2class_hate_speech']].groupby('hate_sp
df = grouped[['hate_speech_score', 'text', '2class_hate_speech']]
df
```

	hate_speech_score	text	2class_hate_speech
0	0	their karma has caught up with them. the bite...	0
1	0	enjoy you're freedom this could be a muslim co...	0
2	0	imagine them existing they would be like 9/10 ...	0
3	0	we will do better than that we will prosecute...	0
4	0	if you gotta press a nigga to do simple shit l...	0
...	...	...	...
26995	2	no huge differences between us and the others ...	0
26996	2	find you a good general contractor he should h...	0
26997	2	that's why that one came here, went through th...	0

Train/Test/Val Split

```
np.random.seed(6543)
shuffle = np.random.permutation(np.arange(df.shape[0]))

train_data, val_data, test_data = df.iloc[shuffle][:round(len(df)*.7)], df.iloc[shuffl

train_examples, train_labels = preprocesser(train_data['text']), np.array(train_data['
val_examples, val_labels = preprocesser(val_data['text']), np.array(val_data['hate_spe
test_examples, test_labels = preprocesser(test_data['text']), np.array(test_data['hate
binary_train_examples, binary_train_labels = preprocesser(train_data['text']), np.arra
binary_val_examples, binary_val_labels = preprocesser(val_data['text']), np.array(val_
binary_test_examples, binary_test_labels = preprocesser(test_data['text']), np.array(t
```

```
train_data['2class_hate_speech'].value_counts()
```

```
0    12554
1     6346
Name: 2class_hate_speech, dtype: int64
```

```
print(len(train_examples) == len(train_labels))
print(len(val_examples) == len(val_labels))
print(len(test_examples) == len(test_labels))
```

```
True
True
True
```

```
print(train_labels[:10])
print(train_data['hate_speech_score'][:10].values)
```

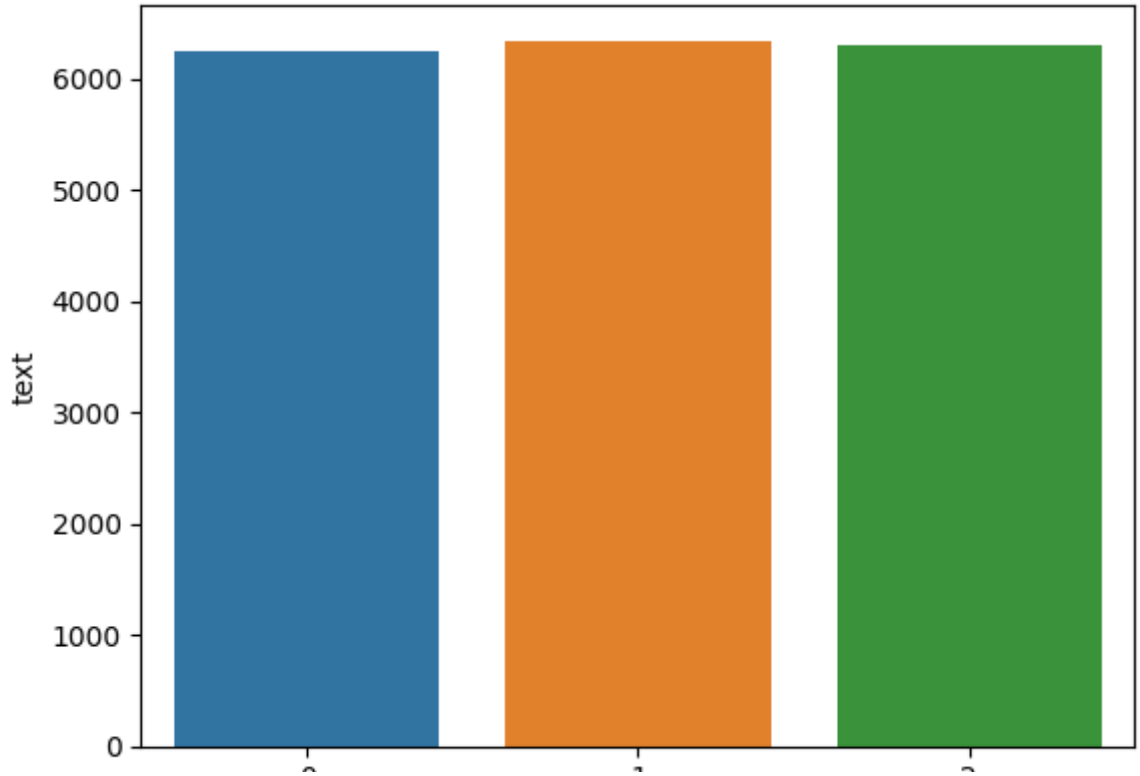
```
[0 2 0 0 2 0 1 0 2 1]
[0 2 0 0 2 0 1 0 2 1]
```

```
val_data['hate_speech_score'].value_counts()
```

```
0    1378
2    1347
1    1325
Name: hate_speech_score, dtype: int64
```

```
labels1 = train_data.groupby('hate_speech_score').agg('count')
sns.barplot(x = labels1.index, y = labels1['text'])
```

```
<Axes: xlabel='hate_speech_score', ylabel='text'>
```



Regular Tokenization

```
tokenizer = tf_text.WhitespaceTokenizer()
train_tokens = tokenizer.tokenize(train_examples)
val_tokens = tokenizer.tokenize(val_examples)
test_tokens = tokenizer.tokenize(test_examples)
```

```
max([len(i) for i in train_tokens.numpy()])
```

```
128
```

```
np.mean([len(i) for i in train_tokens.numpy()])
```

```
24.330529100529102
```

```
np.median([len(i) for i in train_tokens.numpy()])
```

```
19.0
```

Baseline

```
def baseline_model(lbls):
  guesses = []
  for i in range(len(lbls)):
    x = np.random.uniform(low=0, high=1)
    if x <= supp_count/total_count:
      guesses.append(2)
    elif x <= (supp_count + hate_count)/total_count and x > supp_count/total_count:
      guesses.append(1)
    elif x > (supp_count + hate_count)/total_count and x<=1:
      guesses.append(0)
  return guesses


baseline_guesses = baseline_model(test_labels)


baseline_accuracy = np.mean([1 if baseline_guesses[i] == test_labels[i] else 0 for i i
baseline_accuracy
```

```
0.34617283950617284
```

```
print(classification_report(test_labels, baseline_guesses))
```

```
                precision    recall  f1-score   support

             0       0.36      0.25      0.29      1343
             1       0.33      0.32      0.33      1329
             2       0.35      0.46      0.40      1378

      accuracy                           0.35      4050
     macro avg       0.35      0.35      0.34      4050
  weighted avg       0.35      0.35      0.34      4050
```

```
cm = tf.math.confusion_matrix(test_labels, baseline_guesses)
cm = cm/cm.numpy().sum(axis=1)[:, tf.newaxis]


plt.figure(figsize=(20,7))
sns.heatmap(
    cm, annot=True,
    xticklabels=["Neutral", "Hateful", "Supportive"],
    yticklabels=["Neutral", "Hateful", "Supportive"])
plt.xlabel("Predicted")
plt.ylabel("True")
```

```
Text(220.72222222222223, 0.5, 'True')
```

Bert Tokenization

```
#Train and Test for Bert Based Models
bert_tokenizer = BertTokenizer.from_pretrained('bert-base-cased')
bert_model = TFBertModel.from_pretrained('bert-base-cased')

MAX_SEQUENCE_LENGTH = 128

bert_train_tokenized = bert_tokenizer(list(train_examples),
              max_length=MAX_SEQUENCE_LENGTH,
              truncation=True,
              padding='max_length',
              return_tensors='tf')
bert_train_inputs = [bert_train_tokenized.input_ids,
                     bert_train_tokenized.token_type_ids,
                     bert_train_tokenized.attention_mask]
bert_train_labels = np.array(train_labels)

bert_val_tokenized = bert_tokenizer(list(val_examples),
              max_length=MAX_SEQUENCE_LENGTH,
              truncation=True,
              padding='max_length',
              return_tensors='tf')
bert_val_inputs = [bert_val_tokenized.input_ids,
```

```
                    bert_val_tokenized.token_type_ids,
                    bert_val_tokenized.attention_mask]
bert_val_labels = np.array(val_labels)

bert_test_tokenized = bert_tokenizer(list(test_examples),
              max_length=MAX_SEQUENCE_LENGTH,
              truncation=True,
              padding='max_length',
              return_tensors='tf')
bert_test_inputs = [bert_test_tokenized.input_ids,
                    bert_test_tokenized.token_type_ids,
                    bert_test_tokenized.attention_mask]
bert_test_labels = np.array(test_labels)
```

```
Some layers from the model checkpoint at bert-base-cased were not used when init
- This IS expected if you are initializing TFBertModel from the checkpoint of a
- This IS NOT expected if you are initializing TFBertModel from the checkpoint o
All the layers of TFBertModel were initialized from the model checkpoint at bert
If your task is similar to the task the model of the checkpoint was trained on,
```

```
#max(len(x) for x in bert_train_inputs)
np.unique(bert_train_labels[:5000], return_counts =True)
```

```
(array([0, 1, 2]), array([1619, 1719, 1662]))
```

BERT CLS

```
#BERT Base Case
def create_bert_cls_classification_model(max_sequence_length=MAX_SEQUENCE_LENGTH,
                          dropout = 0.3,
                          hidden_size = 100,
                          learning_rate=0.00001):
    """
    Build a  classification model with BERT, where you apply CNN layers  to the BERT
    """



    bert_model.trainable = True
    max_length = MAX_SEQUENCE_LENGTH
    input_ids = tf.keras.layers.Input(shape=(max_length,), dtype=tf.int64, name='input
    token_type_ids = tf.keras.layers.Input(shape=(max_length,), dtype=tf.int64, name=
    attention_mask = tf.keras.layers.Input(shape=(max_length,), dtype=tf.int64, name=

    bert_inputs = {'input_ids': input_ids,
                   'token_type_ids': token_type_ids,
                   'attention_mask': attention_mask}

    bert_out = bert_model(bert_inputs)
```

```
    cls_token = bert_out[0][:, 0, :]

    hidden = tf.keras.layers.Dense(hidden_size, activation='relu', name='hidden_layer'

    hidden = tf.keras.layers.Dropout(dropout)(hidden)

    classification = tf.keras.layers.Dense(3, activation='softmax',name='classificatio

    classification_model = tf.keras.Model(inputs=[input_ids, token_type_ids, attention

    classification_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=lear
                                 loss=tf.keras.losses.SparseCategoricalCrossentropy(fr
                                 metrics='accuracy')

    return classification_model

bert_cls_classification_model = create_bert_cls_classification_model()
bert_cls_classification_model.summary()
#confirm all layers are frozen
bert_cls_classification_model_history = bert_cls_classification_model.fit(
    [bert_train_inputs[0], bert_train_inputs[1], bert_train_inputs[2]],
    bert_train_labels,
    validation_data=([bert_val_inputs[0], bert_val_inputs[1], bert_val_inputs[2]], ber
    batch_size=8,
    epochs=4)
```

```
Model: "model_2"
__________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
==================================================================================
 attention_mask_layer (InputLay  [(None, 128)]       0           []
 er)

 input_ids_layer (InputLayer)   [(None, 128)]        0           []

 token_type_ids_layer (InputLay  [(None, 128)]       0           []
 er)

 tf_bert_model (TFBertModel)    multiple             108310272   ['attention_mask
                                                                  'input_ids_laye
                                                                  'token_type_ids

 tf.__operators__.getitem_2 (Sl  (None, 768)         0           ['tf_bert_model
 icingOpLambda)

 hidden_layer (Dense)           (None, 100)          76900       ['tf.__operators
                                                                 ]']

 dropout_39 (Dropout)           (None, 100)          0           ['hidden_layer[0

 classification_layer (Dense)   (None, 3)            303         ['dropout_39[0]

==================================================================================
```

```
Total params: 108,387,475
Trainable params: 108,387,475
Non-trainable params: 0
_________________________________________________________________
Epoch 1/4
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pool
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pool
2363/2363 [==============================] - 611s 252ms/step - loss: 0.6533 - acc
Epoch 2/4
2363/2363 [==============================] - 591s 250ms/step - loss: 0.5181 - acc
Epoch 3/4
2363/2363 [==============================] - 588s 249ms/step - loss: 0.4496 - acc
Epoch 4/4
2363/2363 [==============================] - 589s 249ms/step - loss: 0.3960 - acc
```

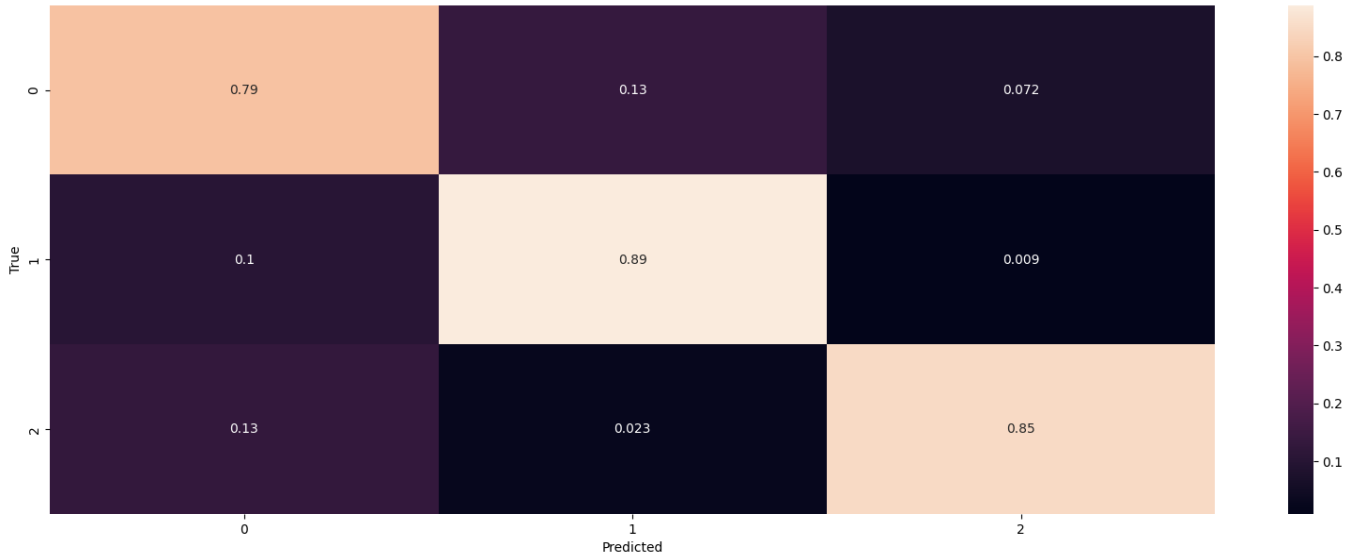
```
cls_predictions = bert_cls_classification_model.predict([bert_test_inputs[0], bert_tes
cls_predictions = tf.argmax(cls_predictions, axis=-1)
```

```
print(classification_report(bert_test_labels, cls_predictions))#, target_names=["Neutr
```

```
                  precision    recall  f1-score   support

               0       0.78      0.84      0.81      1343
               1       0.88      0.88      0.88      1329
               2       0.92      0.87      0.89      1378

        accuracy                           0.86      4050
       macro avg       0.86      0.86      0.86      4050
    weighted avg       0.86      0.86      0.86      4050
```

```
cm = tf.math.confusion_matrix(bert_test_labels, cls_predictions)
cm = cm/cm.numpy().sum(axis=1)[:, tf.newaxis]
plt.figure(figsize=(20,7))
sns.heatmap(
    cm, annot=True)
plt.xlabel("Predicted")
plt.ylabel("True")
```

```
Text(220.72222222222223, 0.5, 'True')
```



Bert CNN

```
def create_bert_cnn_model(max_sequence_length=MAX_SEQUENCE_LENGTH,
                          num_filters = [100, 100, 50, 25],
                          kernel_sizes = [2, 3, 4, 5],
                          dropout = 0.3,
                          hidden_size = 100,
                          learning_rate=0.00001):
    """
    Build a  classification model with BERT, where you apply CNN layers  to the BERT c
    """


    bert_model.trainable = True
    input_ids = tf.keras.layers.Input(shape=(MAX_SEQUENCE_LENGTH,), dtype=tf.int64, na
    token_type_ids = tf.keras.layers.Input(shape=(MAX_SEQUENCE_LENGTH,), dtype=tf.int6
    attention_mask = tf.keras.layers.Input(shape=(MAX_SEQUENCE_LENGTH,), dtype=tf.int6

    bert_inputs = {'input_ids': input_ids,
                   'token_type_ids': token_type_ids,
                   'attention_mask': attention_mask}

    bert_out = bert_model(bert_inputs)[0]

    conv_layers_for_all_kernel_sizes = []
    for kernel_size, filters in zip(kernel_sizes, num_filters):
        conv_layer = keras.layers.Conv1D(filters=filters, kernel_size=kernel_size, act
        conv_layer = keras.layers.GlobalMaxPooling1D()(conv_layer)
        conv_layers_for_all_kernel_sizes.append(conv_layer)
```

```
    conv_output = keras.layers.concatenate(conv_layers_for_all_kernel_sizes, axis=1)

    dropout = keras.layers.Dropout(rate=dropout)(conv_output)

    hidden_output = tf.keras.layers.Dense(hidden_size, activation='relu', name='hidder

    classification = tf.keras.layers.Dense(3, activation='softmax',name='classificatic

    classification_model = tf.keras.Model(inputs=[input_ids, token_type_ids, attentior

    classification_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=lear
                                 loss=tf.keras.losses.SparseCategoricalCrossentropy(fr
                                 metrics='accuracy')


    return classification_model

bert_cnn_model = create_bert_cnn_model()
bert_cnn_model.summary()
#confirm all layers are frozen
bert_cnn_model_history = bert_cnn_model.fit(
    [bert_train_inputs[0], bert_train_inputs[1], bert_train_inputs[2]],
    bert_train_labels,
    validation_data=([bert_val_inputs[0], bert_val_inputs[1], bert_val_inputs[2]], ber
    batch_size=8,
    epochs=5)
```

```
Model: "model_3"
__________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
 attention_mask_layer (InputLay  [(None, 128)]       0           []
 er)

 input_ids_layer (InputLayer)   [(None, 128)]        0           []

 token_type_ids_layer (InputLay  [(None, 128)]       0           []
 er)

 tf_bert_model_2 (TFBertModel)  TFBaseModelOutputWi  108310272   ['attention_mask
                                thPoolingAndCrossAt               'input_ids_laye
                                tentions(last_hidde               'token_type_ids
                                n_state=(None, 128,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 conv1d_12 (Conv1D)             (None, 127, 100)     153700      ['tf_bert_model_
```

```
 conv1d_13 (Conv1D)             (None, 126, 100)     230500      ['tf_bert_model_

 conv1d_14 (Conv1D)             (None, 125, 50)      153650      ['tf_bert_model_

 conv1d_15 (Conv1D)             (None, 124, 25)      96025       ['tf_bert_model_

 global_max_pooling1d_12 (Globa  (None, 100)         0           ['conv1d_12[0][0
 lMaxPooling1D)

 global_max_pooling1d_13 (Globa  (None, 100)         0           ['conv1d_13[0][0
 lMaxPooling1D)

 global_max_pooling1d_14 (Globa  (None, 50)          0           ['conv1d_14[0][0
 lMaxPooling1D)

 global_max_pooling1d_15 (Globa  (None, 25)          0           ['conv1d_15[0][0
 lMaxPooling1D)

 concatenate_3 (Concatenate)    (None, 275)          0           ['global_max_poo
                                                                  'global_max_poo
                                                                  'global_max_poo
                                                                  'global_max_poo

 dropout_114 (Dropout)          (None, 275)          0           ['concatenate_3

 hidden_layer (Dense)           (None, 100)          27600       ['dropout_114[0

 classification_layer (Dense)   (None, 3)            303         ['hidden_layer[0

==================================================================================
Total params: 108,972,050
```

```
np.unique(bert_train_labels, return_counts = True)
```

```
(array([0, 1, 2]), array([6244, 6346, 6310]))
```

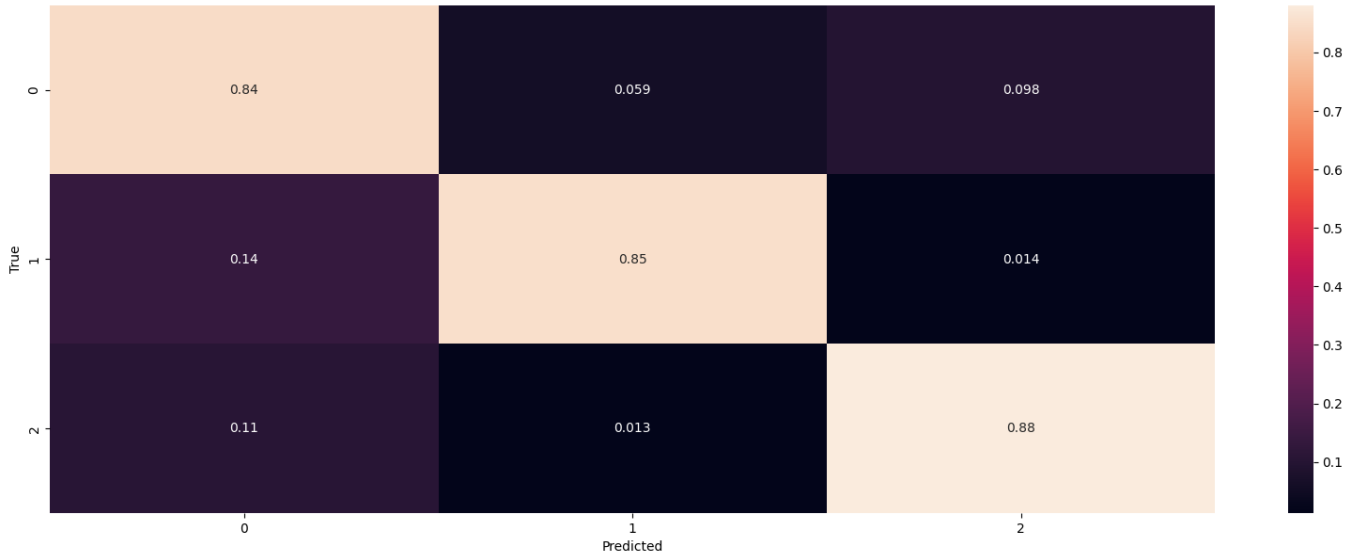
```
cnn_predictions = bert_cnn_model.predict([bert_test_inputs[0], bert_test_inputs[1], be
cnn_predictions = tf.argmax(cnn_predictions, axis=-1)
```

```
print(classification_report(bert_test_labels, cnn_predictions))#, target_names=["Neutr
```

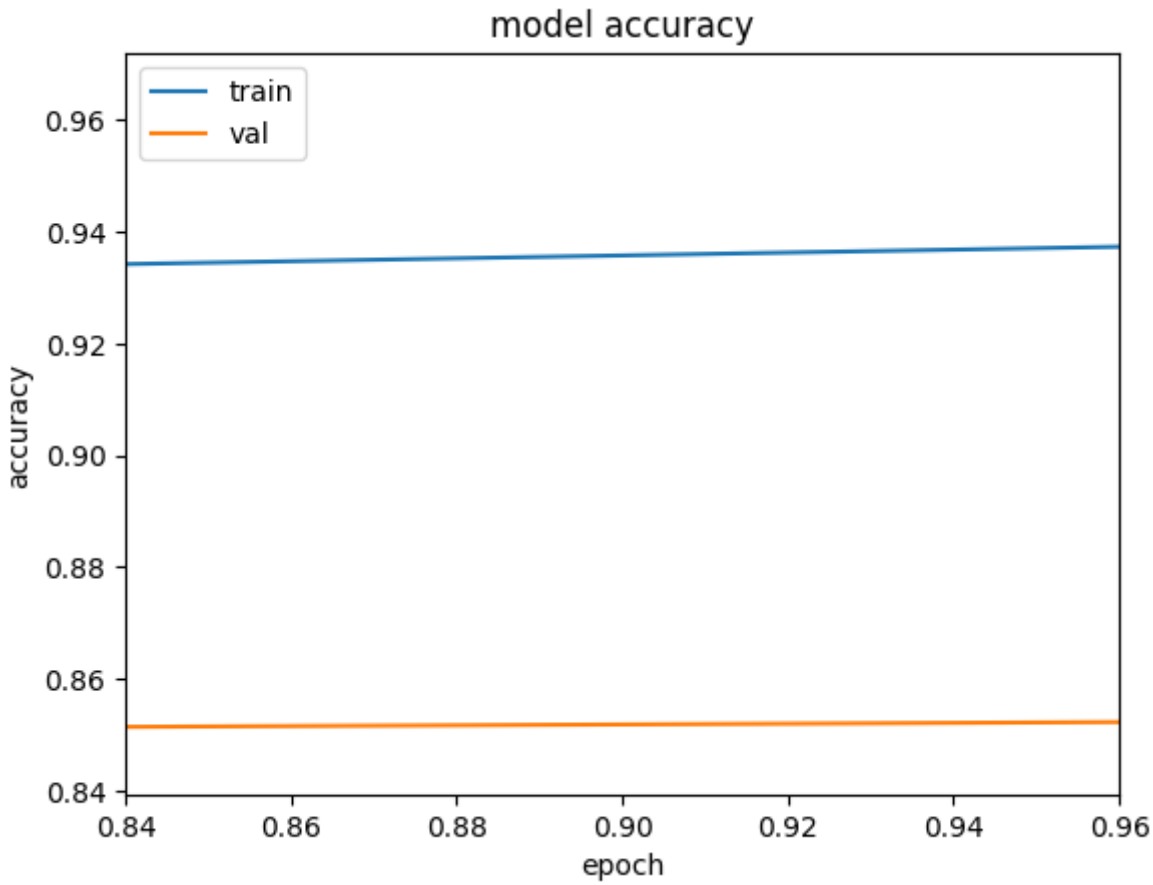
```
                precision    recall  f1-score   support

             0       0.78      0.84      0.81      1343
             1       0.92      0.85      0.88      1329
             2       0.89      0.88      0.89      1378

      accuracy                           0.86      4050
     macro avg       0.86      0.86      0.86      4050
  weighted avg       0.86      0.86      0.86      4050
```

```
cm = tf.math.confusion_matrix(bert_test_labels, cnn_predictions)
cm = cm/cm.numpy().sum(axis=1)[:, tf.newaxis]
plt.figure(figsize=(20,7))
sns.heatmap(
    cm, annot=True)
plt.xlabel("Predicted")
plt.ylabel("True")
```

```
Text(220.72222222222223, 0.5, 'True')
```




```
from matplotlib import pyplot as plt
plt.plot(bert_cnn_model_history.history['accuracy'])
plt.plot(bert_cnn_model_history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.xlim(.84, .96)
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



Roberta

14 cells hidden

Roberta Multiclass, Binary, and Combined

```
from transformers import RobertaConfig, TFRobertaModel


#Roberta Model, Tokenizer, and Train and Test Sets

roberta_tokenizer = AutoTokenizer.from_pretrained("roberta-base")
roberta_model = TFRobertaModel.from_pretrained("roberta-base")

MAX_SEQUENCE_LENGTH = 128
roberta_train_tokenized = roberta_tokenizer(list(train_examples),
              max_length=MAX_SEQUENCE_LENGTH,
              truncation=True,
              padding='max_length',
              add_special_tokens=True,
              return_tensors='tf')
print(roberta_train_tokenized)
```

```
roberta_train_inputs = [roberta_train_tokenized.input_ids,
                        roberta_train_tokenized.attention_mask]
roberta_train_labels = np.array(train_labels)

roberta_val_tokenized = roberta_tokenizer(list(val_examples),
              max_length=MAX_SEQUENCE_LENGTH,
              truncation=True,
              padding='max_length',
              return_tensors='tf')
roberta_val_inputs = [roberta_val_tokenized.input_ids,
                      roberta_val_tokenized.attention_mask]
roberta_val_labels = np.array(val_labels)

roberta_test_tokenized = roberta_tokenizer(list(test_examples),
              max_length=MAX_SEQUENCE_LENGTH,
              truncation=True,
              padding='max_length',
              return_tensors='tf')
roberta_test_inputs = [roberta_test_tokenized.input_ids,
                       #roberta_test_tokenized.token_type_ids,
                       roberta_test_tokenized.attention_mask]
roberta_test_labels = np.array(test_labels)
```

```
Downloading (…)lve/main/config.json:                481/481 [00:00<00:00,
100%                                                 7.55kB/s]
Downloading (…)olve/main/vocab.json:                899k/899k [00:00<00:00,
100%                                                 3.71MB/s]
Downloading (…)olve/main/merges.txt:                456k/456k [00:00<00:00,
100%                                                 5.25MB/s]
Downloading                                         1.36M/1.36M [00:00<00:00,
(…)/main/tokenizer.json: 100%                        9.46MB/s]
Downloading tf_model.h5:                            657M/657M [00:05<00:00,
100%                                                 146MB/s]
Some layers from the model checkpoint at roberta-base were not used when initial
- This IS expected if you are initializing TFRobertaModel from the checkpoint of
- This IS NOT expected if you are initializing TFRobertaModel from the checkpoin
All the layers of TFRobertaModel were initialized from the model checkpoint at r
If your task is similar to the task the model of the checkpoint was trained on, y
{'input_ids': <tf.Tensor: shape=(18900, 128), dtype=int32, numpy=
array([[    0, 24916,   324, ...,     1,     1,     1],
       [    0,   118,   657, ...,     1,     1,     1],
       [    0,  1694,  1595, ...,     1,     1,     1],
       ...,
       [    0,   113,   428, ...,     1,     1,     1],
       [    0, 38060,   197, ...,     1,     1,     1],
       [    0,  3999,   102, ...,     1,     1,     1]], dtype=int32)>, 'attentio
array([[1, 1, 1, ..., 0, 0, 0],
```

```
def create_roberta1_classification_model(max_sequence_length=MAX_SEQUENCE_LENGTH,
                         num_filters = [100, 100, 50, 25],
                         kernel_sizes = [2, 3, 4, 5],
                         dropout = 0.3,
                         hidden_size = 100,
                         learning_rate=.00001):
    """
    Build a  classification model with BERT, where you apply CNN layers  to the BERT c
    """



    roberta_model.trainable = True
    #max_length = 100
    input_ids = tf.keras.layers.Input(shape=(max_sequence_length,), dtype=tf.int64, na
   # token_type_ids = tf.keras.layers.Input(shape=(max_sequence_length,), dtype=tf.int
    attention_mask = tf.keras.layers.Input(shape=(max_sequence_length,), dtype=tf.int6

    roberta_inputs = {'input_ids': input_ids,
                  'attention_mask': attention_mask}

    roberta_out = roberta_model(roberta_inputs)

    pooler_token = roberta_out[1]
    print(pooler_token)
    hidden = tf.keras.layers.Dense(hidden_size, activation='relu', name='hidden_layer'

    hidden = tf.keras.layers.Dropout(dropout)(hidden)

    classification = tf.keras.layers.Dense(3, activation='softmax',name='classificatio

    classification_model = tf.keras.Model(inputs=[input_ids, attention_mask], outputs=

    classification_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=lear
                                 loss=tf.keras.losses.SparseCategoricalCrossentropy(fr
                                 metrics='accuracy')

    return classification_model


roberta1_classification_model = create_roberta1_classification_model()
roberta1_classification_model.summary()
#confirm all layers are frozen
roberta1_classification_model_history = roberta1_classification_model.fit(
    [roberta_train_inputs[0], roberta_train_inputs[1]],
    roberta_train_labels,
    validation_data=([roberta_val_inputs[0], roberta_val_inputs[1]], roberta_val_label
    batch_size=8,
    epochs=8)
```

```
KerasTensor(type_spec=TensorSpec(shape=(None, 768), dtype=tf.float32, name=None)
Model: "model"
```

```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
 attention_mask_layer (InputLay  [(None, 128)]       0           []
 er)

 input_ids_layer (InputLayer)   [(None, 128)]        0           []

 tf_roberta_model (TFRobertaMod  TFBaseModelOutputWi  124645632  ['attention_mask
 el)                            thPoolingAndCrossAt               'input_ids_laye
                                ttentions(last_hidde
                                n_state=(None, 128,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 hidden_layer (Dense)           (None, 100)          76900       ['tf_roberta_mod

 dropout_37 (Dropout)           (None, 100)          0           ['hidden_layer[0

 classification_layer (Dense)   (None, 3)            303         ['dropout_37[0]

==================================================================================
Total params: 124,722,835
Trainable params: 124,722,835
Non-trainable params: 0
__________________________________________________________________________________
Epoch 1/8
2363/2363 [==============================] - 650s 267ms/step - loss: 1.1776 - acc
Epoch 2/8
2363/2363 [==============================] - 626s 265ms/step - loss: 1.0371 - acc
Epoch 3/8
2363/2363 [==============================] - 626s 265ms/step - loss: 0.9126 - acc
Epoch 4/8
2363/2363 [==============================] - 626s 265ms/step - loss: 0.7784 - acc
Epoch 5/8
2363/2363 [==============================] - 624s 264ms/step - loss: 0.6492 - acc
Epoch 6/8
2363/2363 [==============================] - 622s 263ms/step - loss: 0.5380 - acc
Epoch 7/8
2363/2363 [==============================] - 622s 263ms/step - loss: 0.4488 - acc
Epoch 8/8
2363/2363 [==============================] - 622s 263ms/step - loss: 0.3704 - acc
```

```
predictions1 = roberta1_classification_model.predict([roberta_test_inputs[0], roberta_
```

```
predictions1 = tf.argmax(predictions1, axis=-1)
```

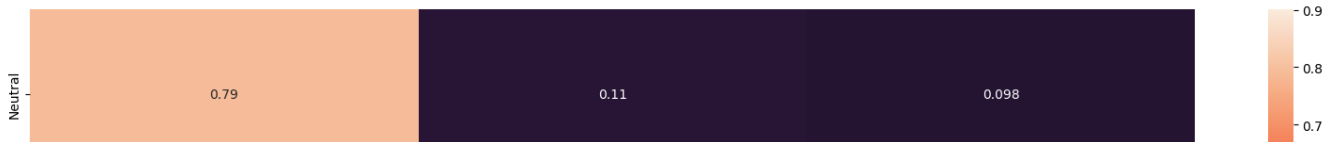
```
from sklearn.metrics import classification_report
print(classification_report(roberta_test_labels, predictions1))#, target_names=["Neutr
```

```
                  precision    recall  f1-score   support

               0       0.83      0.79      0.81      1343
               1       0.87      0.89      0.88      1329
               2       0.89      0.90      0.90      1378

        accuracy                           0.86      4050
       macro avg       0.86      0.86      0.86      4050
    weighted avg       0.86      0.86      0.86      4050
```

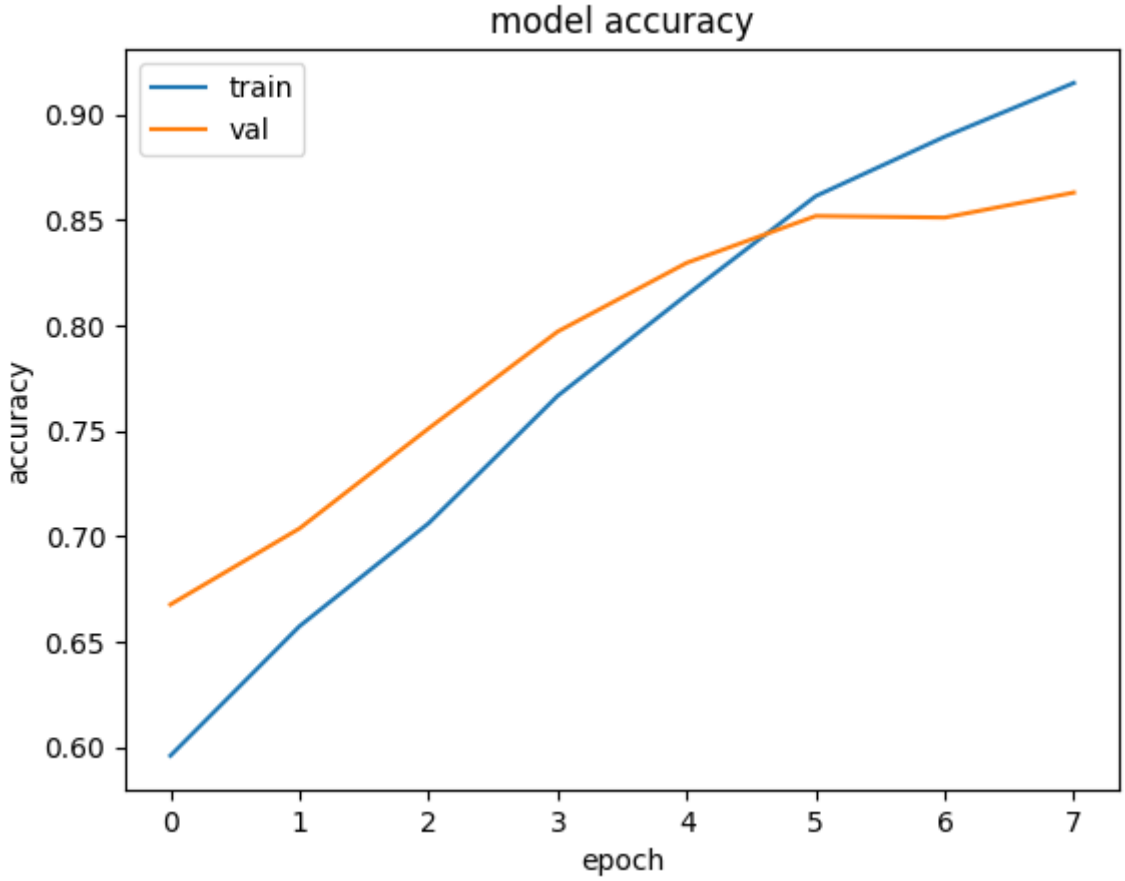
```
cm = tf.math.confusion_matrix(roberta_test_labels, predictions1)
cm = cm/cm.numpy().sum(axis=1)[:, tf.newaxis]
```

```
plt.figure(figsize=(20,7))
sns.heatmap(
    cm, annot=True,
    xticklabels=["Neutral", "Hateful", "Supportive"],
    yticklabels=["Neutral", "Hateful", "Supportive"])
plt.xlabel("Predicted")
plt.ylabel("True")
```

```
Text(220.72222222222223, 0.5, 'True')
```




```
from matplotlib import pyplot as plt
plt.plot(roberta1_classification_model_history.history['accuracy'])
plt.plot(roberta1_classification_model_history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```




```
incorrects_roberta = []
for i in range(len(predictions1)):
  if predictions1[i] != roberta_test_labels[i]:
    incorrects_roberta.append([predictions1[i],roberta_test_labels[i], test_examples[i

print("Roberta Multiclass Model Mislabeled")
for i in range(10):
  print(f'Text {i}: ' , incorrects_roberta[i][2])
  print(f'Predicted Label {i}: ' , incorrects_roberta[i][0].numpy())
  print(f'True Label {i}: ' , incorrects_roberta[i][1])
```

```
Roberta Multiclass Model Mislabeled
Text 0:  the fact that you are a virgin is so hot! i just imagine what it would l
Predicted Label 0:  1
True Label 0:  0
Text 1:  nice way to go calling it the best upcoming movies when its all about w
Predicted Label 1:  1
True Label 1:  0
Text 2:   your mum can go and shag herself?
Predicted Label 2:  1
True Label 2:  0
Text 3:  all these tender white "waah i'm so fragile so you must use your free t
Predicted Label 3:  0
True Label 3:  1
Text 4:  gov. abbott, figures prayer is the answer. no you fucking idiot, making
Predicted Label 4:  2
True Label 4:  0
Text 5:  fyebae turn on everyone rather they do something to her or not. let her
Predicted Label 5:  1
True Label 5:  0
Text 6:  if they wanna turn everyone gay then they should use the same chemicals
Predicted Label 6:  0
True Label 6:  1
Text 7:  seeking wisdom average white people did nothing wrong. you're just weak
Predicted Label 7:  2
True Label 7:  1
Text 8:  it is so rasict because they are asian. i feel bad for the company
Predicted Label 8:  2
True Label 8:  0
Text 9:  >israel has hypnotized the world, may allah awaken the people and help t
Predicted Label 9:  0
True Label 9:  1
```

```
MAX_SEQUENCE_LENGTH = 128
binary_roberta_train_tokenized = roberta_tokenizer(list(binary_train_examples),
              max_length=MAX_SEQUENCE_LENGTH,
              truncation=True,
              padding='max_length',
              return_tensors='tf')
binary_roberta_train_inputs = [binary_roberta_train_tokenized.input_ids,
                     binary_roberta_train_tokenized.attention_mask]
binary_roberta_train_labels = np.array(binary_train_labels)

binary_roberta_val_tokenized = roberta_tokenizer(list(binary_val_examples),
              max_length=MAX_SEQUENCE_LENGTH,
              truncation=True,
              padding='max_length',
              return_tensors='tf')
binary_roberta_val_inputs = [binary_roberta_val_tokenized.input_ids,
                     binary_roberta_val_tokenized.attention_mask]
binary_roberta_val_labels = np.array(binary_val_labels)

binary_roberta_test_tokenized = roberta_tokenizer(list(binary_test_examples),
              max_length=MAX_SEQUENCE_LENGTH,
```

```
                truncation=True,
                padding='max_length',
                return_tensors='tf')
binary_roberta_test_inputs = [binary_roberta_test_tokenized.input_ids,
                      binary_roberta_test_tokenized.attention_mask]
binary_roberta_test_labels = np.array(binary_test_labels)


def create_roberta2_classification_model(max_sequence_length=MAX_SEQUENCE_LENGTH,
                           num_classes = 1,
                           dropout = 0.3,
                           hidden_size = 100,
                           learning_rate=0.00001):
    """
    Build a  classification model with BERT, where you apply CNN layers  to the BERT c
    """



    roberta_model.trainable = True
    #max_length = 100
    input_ids = tf.keras.layers.Input(shape=(max_sequence_length,), dtype=tf.int64, na
   # token_type_ids = tf.keras.layers.Input(shape=(max_sequence_length,), dtype=tf.int
    attention_mask = tf.keras.layers.Input(shape=(max_sequence_length,), dtype=tf.int6

    roberta_inputs = {'input_ids': input_ids,
                   'attention_mask': attention_mask}

    roberta_out = roberta_model(roberta_inputs)

    cls_token = roberta_out[1]

    hidden = tf.keras.layers.Dense(hidden_size, activation='relu', name='hidden_layer'

    hidden = tf.keras.layers.Dropout(dropout)(hidden)

    classification = tf.keras.layers.Dense(num_classes, activation='sigmoid',name='cla

    classification_model = tf.keras.Model(inputs=[input_ids, attention_mask], outputs=

    classification_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=lear
                                 loss=tf.keras.losses.BinaryCrossentropy(from_logits=F
                                 metrics='accuracy')

    return classification_model


binary_roberta2_classification_model = create_roberta2_classification_model()
binary_roberta2_classification_model.summary()
#confirm all layers are frozen
binary_roberta2_classification_model_history = binary_roberta2_classification_model.fi
    [binary_roberta_train_inputs[0], binary_roberta_train_inputs[1]],
    binary_roberta_train_labels,
```

```
validation_data=([binary_roberta_val_inputs[0], binary_roberta_val_inputs[1]], bir
batch_size=8,
epochs=4)
```

```
Model: "model_1"
__________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
 attention_mask_layer (InputLay  [(None, 128)]       0           []
 er)

 input_ids_layer (InputLayer)   [(None, 128)]        0           []

 tf_roberta_model (TFRobertaMod  TFBaseModelOutputWi  124645632  ['attention_mask
 el)                            thPoolingAndCrossAt               'input_ids_laye
                                ttentions(last_hidde
                                n_state=(None, 128,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 hidden_layer (Dense)           (None, 100)          76900       ['tf_roberta_mod

 dropout_38 (Dropout)           (None, 100)          0           ['hidden_layer[0

 classification_layer (Dense)   (None, 1)            101         ['dropout_38[0]

==================================================================================
Total params: 124,722,633
Trainable params: 124,722,633
Non-trainable params: 0
__________________________________________________________________________________
Epoch 1/3
2363/2363 [==============================] - 643s 265ms/step - loss: 0.2582 - acc
Epoch 2/3
2363/2363 [==============================] - 623s 264ms/step - loss: 0.2092 - acc
Epoch 3/3
2363/2363 [==============================] - 624s 264ms/step - loss: 0.1798 - acc
```

```
binary_predictions2 = binary_roberta2_classification_model.predict([binary_roberta_tes
```

```
binary_predictions2 = [1 if i[0] > .5 else 0 for i in binary_predictions2]
```

```
print(classification_report(binary_roberta_test_labels, binary_predictions2))#, target
```

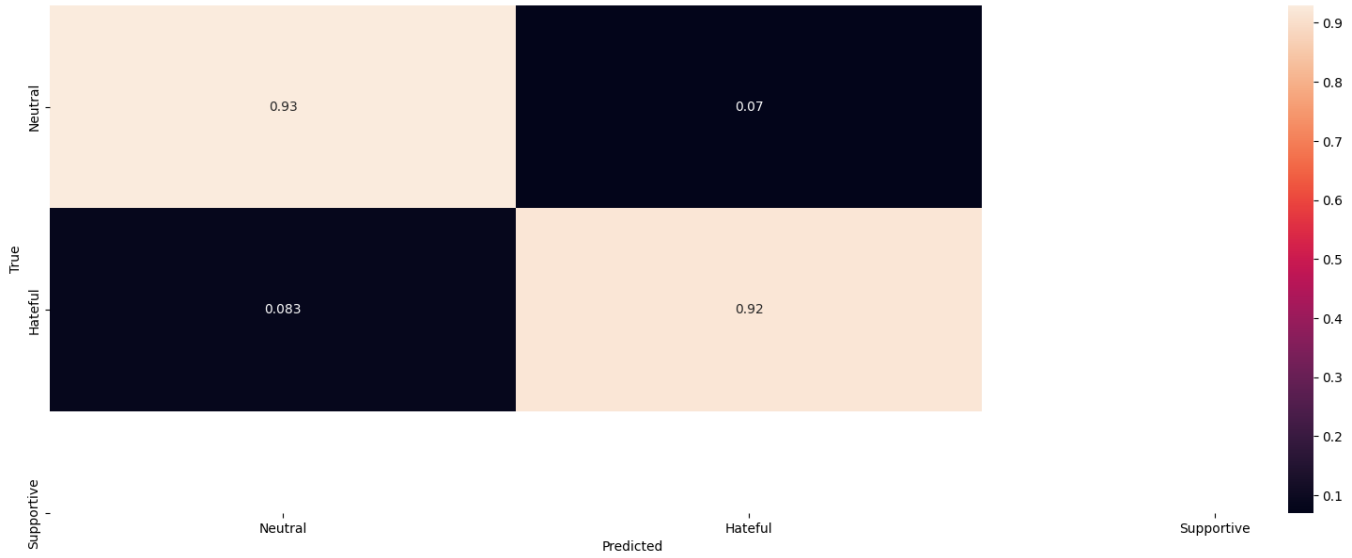
```
              precision    recall  f1-score   support

           0       0.96      0.93      0.94      2721
```

```
                1       0.86      0.92      0.89      1329

         accuracy                           0.93      4050
        macro avg       0.91      0.92      0.92      4050
     weighted avg       0.93      0.93      0.93      4050
```

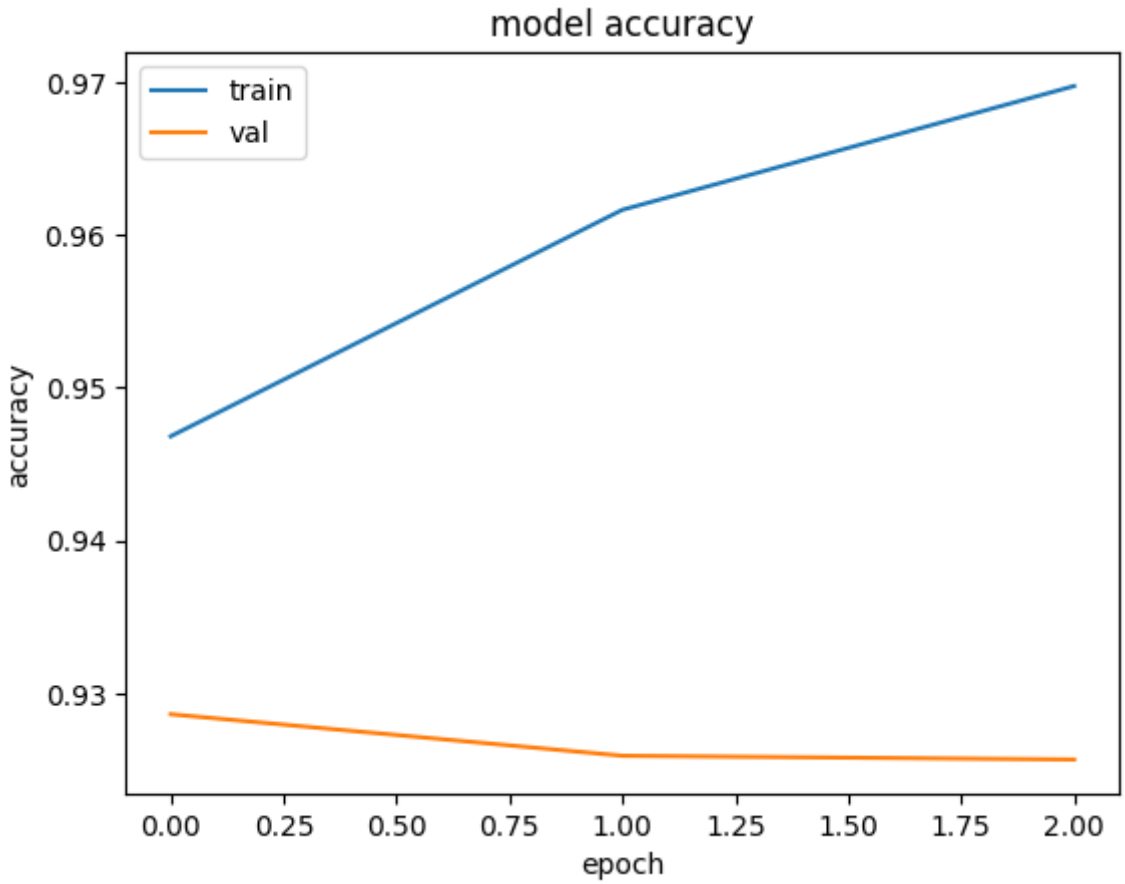
```
cm = tf.math.confusion_matrix(binary_roberta_test_labels, binary_predictions2)
cm = cm/cm.numpy().sum(axis=1)[:, tf.newaxis]
```

```
plt.figure(figsize=(20,7))
sns.heatmap(
    cm, annot=True,
    xticklabels=["Neutral", "Hateful", "Supportive"],
    yticklabels=["Neutral", "Hateful", "Supportive"])
plt.xlabel("Predicted")
plt.ylabel("True")
```

```
Text(220.72222222222223, 0.5, 'True')
```



```
from matplotlib import pyplot as plt
plt.plot(binary_roberta2_classification_model_history.history['accuracy'])
plt.plot(binary_roberta2_classification_model_history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```




```
combined_preds = []
for i in range(len(predictions1)):
  if predictions1[i] == 0 and binary_predictions2[i] == 1:
    combined_preds.append(binary_predictions2[i])
  else:
    combined_preds.append(predictions1[i])
combined_preds = np.array(combined_preds)


print(len(predictions1))
print(len(binary_predictions2))
```

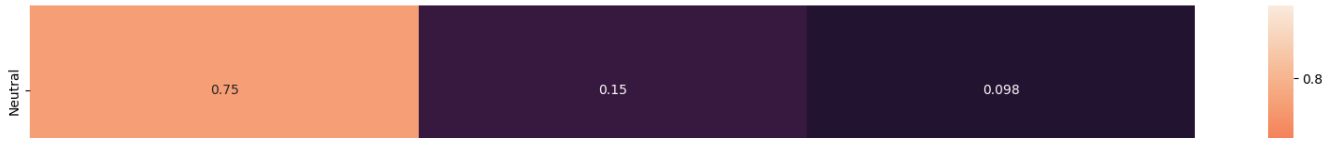
```
4050
4050
```

```
print(classification_report(roberta_test_labels, combined_preds))
```

	precision	recall	f1-score	support
0	0.86	0.75	0.80	1343
1	0.83	0.93	0.88	1329
2	0.89	0.90	0.90	1378
accuracy			0.86	4050
macro avg	0.86	0.86	0.86	4050
weighted avg	0.86	0.86	0.86	4050

```
cm = tf.math.confusion_matrix(roberta_test_labels, combined_preds)
cm = cm/cm.numpy().sum(axis=1)[:, tf.newaxis]


plt.figure(figsize=(20,7))
sns.heatmap(
    cm, annot=True,
    xticklabels=["Neutral", "Hateful", "Supportive"],
    yticklabels=["Neutral", "Hateful", "Supportive"])
plt.xlabel("Predicted")
plt.ylabel("True")
```

```
Text(220.72222222222223, 0.5, 'True')
```




```
incorrects = []
for i in range(len(combined_preds)):
  if combined_preds[i] != roberta_test_labels[i]:
    incorrects.append([combined_preds[i],roberta_test_labels[i], test_examples[i]])
```

```
print("Combined Model Mislabeled")
for i in range(10):
  print(f'Text {i}: ' , incorrects[i][2])
  print(f'Predicted Label {i}: ' , incorrects[i][0])
  print(f'True Label {i}: ' , incorrects[i][1])
```

```
Text 0:  the fact that you are a virgin is so hot! i just imagine what it would l
Predicted Label 0:  1
True Label 0:  0
Text 1:  fuck the fuckboys
Predicted Label 1:  1
True Label 1:  0
Text 2:  nice way to go calling it the best upcoming movies when its all about w
Predicted Label 2:  1
True Label 2:  0
Text 3:   your mum can go and shag herself?
Predicted Label 3:  1
True Label 3:  0
Text 4:  all these tender white "waah i'm so fragile so you must use your free t
Predicted Label 4:  0
True Label 4:  1
Text 5:  gov. abbott, figures prayer is the answer. no you fucking idiot, making
Predicted Label 5:  2
True Label 5:  0
Text 6:  fyebae turn on everyone rather they do something to her or not. let her
Predicted Label 6:  1
True Label 6:  0
Text 7:  sorry but perverts existed already and hasnt increased as proven by me
Predicted Label 7:  1
True Label 7:  0
Text 8:  seeking wisdom average white people did nothing wrong. you're just weak
Predicted Label 8:  2
True Label 8:  1
Text 9:  it is so rasict because they are asian. i feel bad for the company
Predicted Label 9:  2
True Label 9:  0
```