National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Prathamesh Gaikar
Student ID: x19216301

School of Computing
National College of Ireland

Supervisor: Dr. Bharathi Chakravarthi

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Prathamesh Gaikar
Student ID:	x19216301
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Bharathi Chakravarthi
Submission Due Date:	16/08/2021
Project Title:	Configuration Manual
Word Count:	1247
Page Count:	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	23rd September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

List of Figures

Configuration Manual

Prathamesh Gaikar
x19216301

1 Introduction

The step by step process involved in this project from environment setup to implementation and evaluation is discussed in this configuration manual which is to identify whether the performance of transformer based model is improved by applying aspect based sentiment analysis technique on it. The information about the programming language used, configuration of system and necessary libraries are included in this configuration manual. Results of this research along with the different experiments and their evaluation metrics used in this research are discussed.

2 Environment Setup

2.1 System Specification

Implementation of this research was performed on Google Collaboratory. It is online platform build upon Jupyter notebook and its is free. It allocates to run python programs on google servers and influences high end GPU's free of cost to implement machine learning models. Due to faster GPU, the waiting time is less while the code is running. Also, huge setup does not needs to installed in your system for the execution of project and sharing notebooks is very easy. End to End implementation of project can be performed on google colab platform.

2.2 Technical Specification

Python programming language was used for the implementation of this project.For execution of the project, following packages were used:

- NumPy
- Matplotlib
- Pandas
- transformers
- Keras
- pytorch
- tensorflow

- seaborn
- sentenccepiece
- Scikit-learn

2.3 Data Source

The data is gathered from official website of Amazon web service. The data was downloaded in .tsv format. The dataset consists of 1705837 records. The data has various parameters such as customer id, review id, product id , review body, review title, star ratings, product parent, product title, product category, helpful votes, total votes, vine, review headline, review date. The dataset has reviews of different products from the year 2004 - 2015. To achieve the objective and goal of the project, the data has to be polished, and models should be trained on data.

3 Implementation

In this section end to end steps are elaborated while performing the implementation proposed research. The implementation consists of following steps: Data Preparation and data pre-processing and implementation of models.

3.1 Importing Necessary Libraries

In figure 1 all the necessary libraries which are required for implementation of this project are imported

```
import numpy as np
import regex as re
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statistics
import math
import os
import tensorflow as tf

from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

import tensorflow as tf
import tensorflow.keras.backend as K
# from tf.keras.preprocessing.text import tokenizers
from transformers import RobertaTokenizer, TFRobertaModel

from collections import Counter

import warnings
warnings.filterwarnings("ignore")
```

Figure 1: Necessary libraries

3.2 Data Loading and Pre-processing

Fig 2 shows the original data

	marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes	total_votes	vine	verified_purchase	review_headline	review_body	review_date
0	UK	10349	R2YVNBBMXD8KVJ	B00MWK7BWG	307651059	My Favourite Faded Fantasy	Music	5	0	0	N	Y	Five Stars	The best album ever!	2014-12-29
1	UK	10629	R2K4BOL8MN1TTY	B006CHML4I	835010224	Seiko 5 Men's Automatic Watch with Black Dial ...	Watches	4	0	0	N	Y	Great watch from casio.	What a great watch. Both watches and strap is ...	2013-10-24
2	UK	12136	R3P40IEALROVCH	B00IIFCJX0	271687675	Dexter Season 8	Digital_Video_Download	5	0	0	N	Y	fantastic	love watching all the episodes of Dexter, when...	2014-05-09
3	UK	12268	R25XL1WWYRDLA9	B000W7JWUA	211383699	The Settlers of Catan Board Game - discontinue...	Toys	5	0	0	N	Y	Five Stars	Excellent game!!!	2014-09-19
4	UK	12677	RVTVB9YDXSFYH	B005JTAP4S	182965893	Peter: A Darkened Fairytale (Vol 1)	Digital_Ebook_Purchase	5	12	12	N	N	A twist on Tales	This cute, quick read is very different to say...	2013-09-18

Figure 2: Loading of data

Since the dataset has multiple languages and for extracting of aspect terms spaCy library is to be used trained on english module. So, the languages other than english has to be removed from the dataset which is done by filtering the english reviews using langdetect. The code for filtering the language is shown in fig 3 and reviewlanguage column has been added to dataset which is reflecting in figure 4.

```
dfLang.reviewLanguage.value_counts()
dfEn = dfLang[dfLang["reviewLanguage"] == "en"]
# filtering out the english reviews
dfEn.head()
```

Figure 3: Filtering english language code

marketplace	customer_id	review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes	total_votes	vine	verified_purchase	review_headline	review_body	review_date	reviewLanguage
UK	10349	R2YVNBBMXD8KVJ	B00MWK7BWG	307651059	My Favourite Faded Fantasy	Music	5	0	0	N	Y	Five Stars	The best album ever!	2014-12-29	en
UK	10629	R2K4BOL8MN1TTY	B006CHML4I	835010224	Seiko 5 Men's Automatic Watch with Black Dial ...	Watches	4	0	0	N	Y	Great watch from casio.	What a great watch. Both watches and strap is ...	2013-10-24	en
UK	12136	R3P40IEALROVCH	B00IIFCJX0	271687675	Dexter Season 8	Digital_Video_Download	5	0	0	N	Y	fantastic	love watching all the episodes of Dexter, when...	2014-05-09	en
UK	12677	RVTVB9YDXSFYH	B005JTAP4S	182965893	Peter: A Darkened Fairytale (Vol 1)	Digital_Ebook_Purchase	5	12	12	N	N	A twist on Tales	This cute, quick read is very different to say...	2013-09-18	en
UK	13070	R1P16QCZR7RHM	B00004WMYB	530484605	The Marshall Mathers LP	Music	1	1	7	N	Y	scratches n a crack	im very disappointed in amazon, theyre startin...	2013-07-30	en

Figure 4: Filtered English languages

3.3 Aspect Extraction

After the removal of reviews in other than english language, the process of aspect extraction begins. In fig 5 Code for extracting aspect has been defined. For aspect extraction, spaCy library is used. Here after extracting the aspect terms and calculating the

sentiments as positive or negative, a column has been added to the data of aspectAverageScore as shown in figure 8. In this data, for calculating aspectAverageScore first the aspect terms and the sentiments associated with it are extracted. If there are more than 1 aspect and associated sentiment, then the average of the sentiment is taken. For example. The book title is good, but the stories are bad and also the pictures are not good; In this example book, stories and pictures are the aspects and there are associated sentiments with it. So according to the average of the sentiment it is marked as negative sentiment i.e., 0 in aspectAverageScore column in the dataset.

```
import spacy
nlp = spacy.load("en_core_web_sm")
```

```
from tqdm import tqdm
from textblob import TextBlob
```

```
def aspectSentiment(text):
  averageSentiment = 0
  count = 0
  aspects = []
  doc = nlp(text)
  descriptive_term = ''
  target = ''
  for token in doc:
    if token.dep_ == 'nsubj' and token.pos_ == 'NOUN': # extracting the aspects
      target = token.text
    if token.pos_ == 'ADJ':
      prepend = ''
      for child in token.children:
        if child.pos_ != 'ADV':
          continue
        prepend += child.text + ' '
      descriptive_term = prepend + token.text

  aspects.append({'aspect': target,
    'description': descriptive_term})

  for aspect in aspects:
    averageSentiment += TextBlob(aspect['description']).sentiment[0] # checking the sentiments using textblob
    count += 1
    # aspect['sentiment'] = TextBlob(aspect['description']).sentiment

  if averageSentiment/count > 0:
    return 1
  else:
    return 0
```

Figure 5: Aspect extraction

```
dfSample["aspectAverageScore"] = dfSample.review_body.progress_apply(lambda x: aspectSentiment(x))
# applying the aspectsentiment function
```

```
100%|██████████| 1000000/1000000 [5:41:52<00:00, 48.75it/s]
```

Figure 6: Aspect Average score

In figure 7 data is labelled properly and it provides ground truth that the machine learning algorithm uses to check its prediction accuracy. In this data has been labelled on the

basis of star ratings. star ratings have values from 1 to 5. So, whenever the star ratings value is above 3 then it is considered as positive sentiment and whenever the star rating is lower or equal to 3 then it is considered as negative sentiment. The positive sentiment is denoted as '1' and the negative sentiment is denoted as '0'

```
[ ]  dfPlot['sentiment'] = dfPlot.star_rating.apply(lambda x: 1 if x >3 else 0)
```

Figure 7: labelling data on the basis of star ratings

Here, it can be seen that both sentiment and aspectaveragescore data has been added to dataset.

review_id	product_id	product_parent	product_title	product_category	star_rating	helpful_votes	total_votes	vine	verified_purchase	review_headline	review_body	review_date	reviewLanguage	aspectAverageScore	sentiment
KFFX4PWF3RP8	B00B0X4PIO	536757221	And the Mountains Echoed	Digital_Ebook_Purchase	2	0	0	N	Y	Disappointing	Very disappointing. I was really looking forwa...	2013-11-05	en	0	0
4TKA3VZ2AM5D	B005KIR3PC	581090644	LEGO Creator 6913: Blue Roadster	Toys	5	0	0	N	Y	Lego at it best.	I bought it as a gift for my grandaughter who ...	2013-01-15	en	1	1
23K466JZSQH9C	B001BJARYI	361884839	Lost - Season 4 [DVD]	Video DVD	5	0	0	N	Y	Best season so far.	This is the best season so far with all the di...	2014-03-21	en	0	1
8KVSX9Y103UFZ	1780330960	582342907	A Visit From the Goon Squad	Books	3	1	2	N	N	Imaginative and inventive but.....	Jennifer Egan writes with great energy and ver...	2011-11-21	en	1	0
YVM2AC7CHV5M	0141345659	582629146	The Fault in Our Stars	Books	3	0	0	N	N	I liked it, I just didn't LOVE it...	So having heard that this book had made young ...	2014-06-03	en	1	0

Figure 8: addition of aspect average score and labelled sentiments to the data

3.4 Modelling and Evaluation

The process of data modeling is an act of training machine learning model to predict the values from the features and adjusting it according to the business needs. Various transformer based deep learning method are going to be used such as XLNet, ALBERT and RoBERTa. For modelling only 40000 data is used due to computational limitations.

3.4.1 Splitting the data into train and test

The dataset is split into train and test set with the split ratio of 0.3 for training and validation of the model

Split all data

```
[ ]  tr_inputs, val_inputs, tr_tags, val_tags,tr_masks, val_masks,tr_segs, val_segs = train_test_split(full_input_ids, tags,full_input_masks,full_segment_ids,
                                                                                                       random_state=4, test_size=0.3)

[ ]  len(tr_inputs),len(val_inputs),len(tr_segs),len(val_segs)

     (28000, 12000, 28000, 12000)
```

Figure 9: Split into Train and Test

3.4.2 XLNet

In this paper Alshahrani et al. (2020) multiple models are built on top of XLNet for prediction of cynicism and optimism on twitter messages and positive emotions are much more common in optimistic messages while in pessimistic messages negative emotions are more is demonstrated using XLNet sentiment analysis.

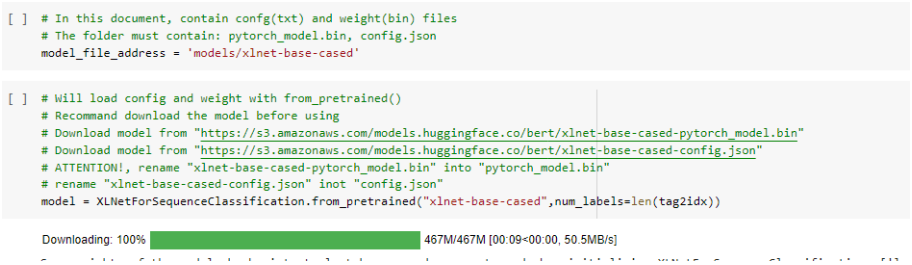

Figure 10: Loading XLNet model

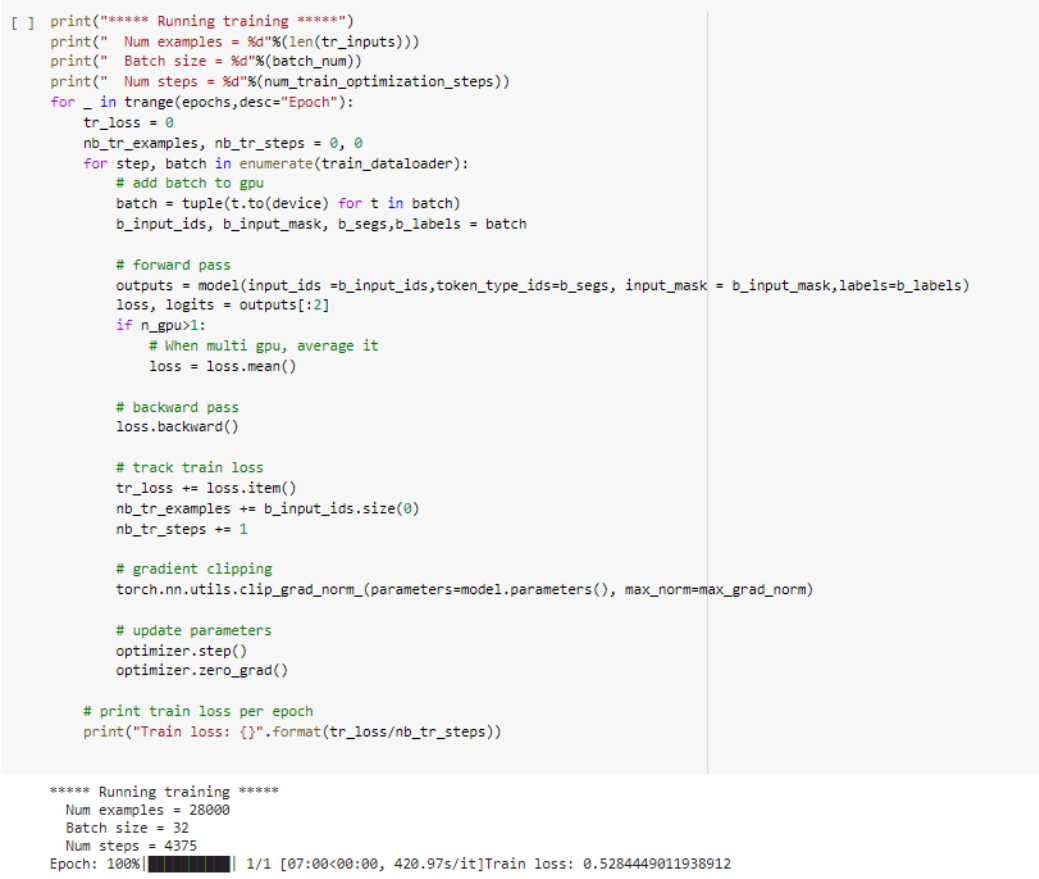

Figure 11: Training of XLNet

```
eval_loss, eval_accuracy = 0, 0
nb_eval_steps, nb_eval_examples = 0, 0

y_true = []
y_predict = []
print("***** Running evaluation *****")
print("  Num examples ={}".format(len(val_inputs)))
print("  Batch size = {}".format(batch_num))
for step, batch in enumerate(valid_dataloader):
    batch = tuple(t.to(device) for t in batch)
    b_input_ids, b_input_mask, b_segs,b_labels = batch

    with torch.no_grad():
        outputs = model(input_ids =b_input_ids,token_type_ids=b_segs, input_mask = b_input_mask,labels=b_labels)
        tmp_eval_loss, logits = outputs[:2]

    # Get textclassification predict result
    logits = logits.detach().cpu().numpy()
    label_ids = b_labels.to('cpu').numpy()
    tmp_eval_accuracy = accuracy(logits, label_ids)
#     print(tmp_eval_accuracy)
#     print(np.argmax(logits, axis=1))
#     print(label_ids)

    # Save predict and real label reuslt for analyze
    for predict in np.argmax(logits, axis=1):
        y_predict.append(predict)

    for real_result in label_ids.tolist():
        y_true.append(real_result)


    eval_loss += tmp_eval_loss.mean().item()
    eval_accuracy += tmp_eval_accuracy

    nb_eval_steps += 1


eval_loss = eval_loss / nb_eval_steps
eval_accuracy = eval_accuracy / len(val_inputs)
loss = tr_loss/nb_tr_steps
result = {'eval_loss': eval_loss,
                'eval_accuracy_sentiment': eval_accuracy,
                'loss': loss}
report = classification_report(y_pred=np.array(y_predict),y_true=np.array(y_true))

# Save the report into file
output_eval_file = os.path.join(xlnet_out_address, "eval_results.txt")
with open(output_eval_file, "w") as writer:
    print("***** Eval results *****")
    for key in sorted(result.keys()):
        print("  %s = %s"%(key, str(result[key])))
        writer.write("%s = %s\n" % (key, str(result[key])))

    print(report)
    writer.write("\n\n")
    writer.write(report)
```

Figure 12: Evaluation of XLNet without aspect terms

```
***** Running evaluation *****
  Num examples =12000
  Batch size = 32
***** Eval results *****
  eval_accuracy_sentiment = 0.9115
  eval_loss = 0.2992655627094209
  loss = 0.19236567653928485
              precision    recall  f1-score   support

           0       0.82      0.58      0.68      1938
           1       0.92      0.98      0.95     10062

    accuracy                           0.91     12000
   macro avg       0.87      0.78      0.81     12000
weighted avg       0.91      0.91      0.91     12000
```

Figure 13: Accuracy of XLNet without aspect terms

After evaluating XLNet model with and without aspect based approach it can be seen that accuracy of XLNet with aspect terms is 79% and accuracy of XLNet without aspect terms is 91%.

```
eval_loss, eval_accuracy = 0, 0
nb_eval_steps, nb_eval_examples = 0, 0

y_true = []
y_predict = []
print("***** Running evaluation *****")
print("  Num examples ={}".format(len(val_inputs)))
print("  Batch size = {}".format(batch_num))
for step, batch in enumerate(valid_dataloader):
    batch = tuple(t.to(device) for t in batch)
    b_input_ids, b_input_mask, b_segs,b_labels = batch

    with torch.no_grad():
        outputs = model(input_ids =b_input_ids,token_type_ids=b_segs, input_mask = b_input_mask,labels=b_labels)
        tmp_eval_loss, logits = outputs[:2]

    # Get textclassification predict result
    logits = logits.detach().cpu().numpy()
    label_ids = b_labels.to('cpu').numpy()
    tmp_eval_accuracy = accuracy(logits, label_ids)
#     print(tmp_eval_accuracy)
#     print(np.argmax(logits, axis=1))
#     print(label_ids)

    # Save predict and real label reuslt for analyze
    for predict in np.argmax(logits, axis=1):
        y_predict.append(predict)

    for real_result in label_ids.tolist():
        y_true.append(real_result)


    eval_loss += tmp_eval_loss.mean().item()
    eval_accuracy += tmp_eval_accuracy

    nb_eval_steps += 1


eval_loss = eval_loss / nb_eval_steps
eval_accuracy = eval_accuracy / len(val_inputs)
loss = tr_loss/nb_tr_steps
result = {'eval_loss': eval_loss,
                'eval_accuracy_aspect': eval_accuracy,
                'loss': loss}
report = classification_report(y_pred=np.array(y_predict),y_true=np.array(y_true))

# Save the report into file
output_eval_file = os.path.join(xlnet_out_address, "eval_results.txt")
with open(output_eval_file, "w") as writer:
    print("***** Eval results *****")
    for key in sorted(result.keys()):
        print("  %s = %s"%(key, str(result[key])))
        writer.write("%s = %s\n" % (key, str(result[key])))

    print(report)
    writer.write("\n\n")
    writer.write(report)
```

Figure 14: Evaluation of XLNet with aspect terms

```
***** Running evaluation *****
  Num examples =12000
  Batch size = 32
***** Eval results *****
  eval_accuracy_aspect = 0.7854166666666667
  eval_loss = 0.4289187043507894
  loss = 0.5284449011938912
              precision    recall  f1-score   support

           0       0.73      0.75      0.74      4861
           1       0.83      0.81      0.82      7139

    accuracy                           0.79     12000
   macro avg       0.78      0.78      0.78     12000
weighted avg       0.79      0.79      0.79     12000
```

Figure 15: Accuracy of XLNet with aspect terms

3.4.3 RoBERTa

In this research Ghasiya and Okamura (2021) the used database has more than a lakh headlines and it was analyzed using word2vec and RoBERTa model. Here, RoBERTa model has accomplished a accuracy of 90% and the headlines were classified better than the traditional classifiers and this execution of RoBERTa for sentiment classification on the dataset showed that the 73.23% of UK news had negative sentiments, while South Korea news has 54.47% of positive sentiment.

```
[ ]  X_train = roberta_encode(X_train, tokenizer)
     X_test = roberta_encode(X_test, tokenizer)

     y_train = np.asarray(y_train, dtype='int32')
     y_test = np.asarray(y_test, dtype='int32')
```

```
[ ]  def build_model(n_categories):
         with strategy.scope():
             input_word_ids = tf.keras.Input(shape=(MAX_LEN,), dtype=tf.int32, name='input_word_ids')
             input_mask = tf.keras.Input(shape=(MAX_LEN,), dtype=tf.int32, name='input_mask')
             input_type_ids = tf.keras.Input(shape=(MAX_LEN,), dtype=tf.int32, name='input_type_ids')

             # Import RoBERTa model from HuggingFace
             roberta_model = TFRobertaModel.from_pretrained(MODEL_NAME)
             x = roberta_model(input_word_ids, attention_mask=input_mask, token_type_ids=input_type_ids)

             # Huggingface transformers have multiple outputs, embeddings are the first one,
             # so let's slice out the first position
             x = x[0]

             x = tf.keras.layers.Dropout(0.1)(x)
             x = tf.keras.layers.Flatten()(x)
             x = tf.keras.layers.Dense(256, activation='relu')(x)
             x = tf.keras.layers.Dense(n_categories, activation='softmax')(x)

             model = tf.keras.Model(inputs=[input_word_ids, input_mask, input_type_ids], outputs=x)
             model.compile(
                 optimizer=tf.keras.optimizers.Adam(lr=1e-5),
                 loss='sparse_categorical_crossentropy',
                 metrics=['accuracy'])

             return model
```

Figure 16: RoBERTa training

The training accuracy is 94% and the testing accuracy is 94%

```
] with strategy.scope():
      print('Training...')
      history = model.fit(X_train,
                          y_train,
                          epochs=2,
                          batch_size=BATCH_SIZE,
                          verbose=1,
                          validation_data=(X_test, y_test))
```

```
Training...
Epoch 1/2
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when calling a model
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kernel:0', 'tf_roberta_model/roberta/
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when calling a model
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kernel:0', 'tf_roberta_model/roberta/
3500/3500 [==============================] - ETA: 0s - loss: 0.1992 - accuracy: 0.9245WARNING:tensorflow:The parameters `output_atte
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
3500/3500 [==============================] - 1884s 536ms/step - loss: 0.1992 - accuracy: 0.9245 - val_loss: 0.1660 - val_accuracy: 0
Epoch 2/2
3500/3500 [==============================] - 1818s 519ms/step - loss: 0.1391 - accuracy: 0.9484 - val_loss: 0.1771 - val_accuracy: 0
```

```
] def plot_confusion_matrix(X_test, y_test, model):
      y_pred = model.predict(X_test)
      y_pred = [np.argmax(i) for i in model.predict(X_test)]

      con_mat = tf.math.confusion_matrix(labels=y_test, predictions=y_pred).numpy()

      con_mat_norm = np.around(con_mat.astype('float') / con_mat.sum(axis=1)[:, np.newaxis], decimals=2)
      label_names = list(range(len(con_mat_norm)))

      con_mat_df = pd.DataFrame(con_mat_norm,
                                index=label_names,
                                columns=label_names)

      figure = plt.figure(figsize=(10, 10))
      sns.heatmap(con_mat_df, cmap=plt.cm.Blues, annot=True)
      plt.ylabel('True label')
      plt.xlabel('Predicted label')
```

```
] scores = model.evaluate(X_test, y_test, verbose=0)
  print("Accuracy: %.2f%%" % (scores[1] * 100))
```

```
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when calling a model
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
Accuracy: 94.00%
```

Figure 17: Evaluation of RoBERTa without aspect terms

The below figure is the confusion matrix on which x-axis is the predicted label and on the y-axis is the Actual label. From the figure it can be seen that the value was negative and it predicted negative for 97% and the value was positive and it was predicted positive 76% of the time. In figure 19 the training accuracy is 88.16% and the testing accuracy is 88.25%

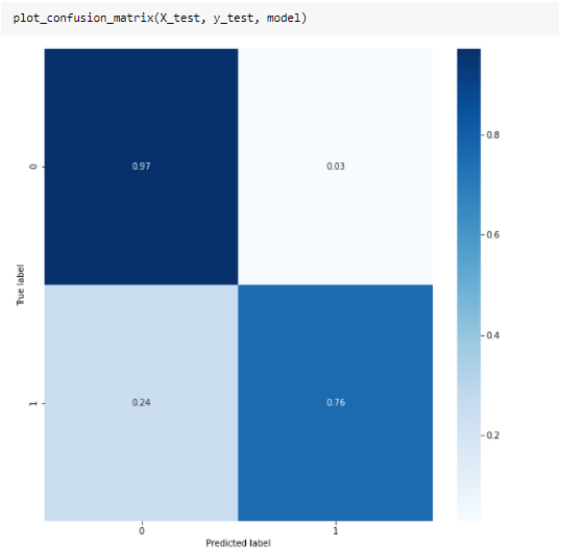

Figure 18: Confusion matrix for RoBERTa without aspect terms

```
with strategy.scope():
    print('Training...')
    history = model.fit(X_train,
                        y_train,
                        epochs=2,
                        batch_size=BATCH_SIZE,
                        verbose=1,
                        validation_data=(X_test, y_test))
```

```
Training...
Epoch 1/2
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when calling a model.They have to be set to True/False
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kernel:0', 'tf_roberta_model/roberta/pooler/dense/bias:0'] when minimi
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when calling a model.They have to be set to True/False
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
WARNING:tensorflow:Gradients do not exist for variables ['tf_roberta_model/roberta/pooler/dense/kernel:0', 'tf_roberta_model/roberta/pooler/dense/bias:0'] when minimi
3000/3000 [==============================] - ETA: 0s - loss: 0.4765 - accuracy: 0.7552WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` ar
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
3000/3000 [==============================] - 1656s 549ms/step - loss: 0.4765 - accuracy: 0.7552 - val_loss: 0.3221 - val_accuracy: 0.8571
Epoch 2/2
3000/3000 [==============================] - 1649s 550ms/step - loss: 0.2753 - accuracy: 0.8816 - val_loss: 0.2740 - val_accuracy: 0.8825
```

```
[ ] def plot_confusion_matrix(X_test, y_test, model):
        y_pred = model.predict(X_test)
        y_pred = [np.argmax(i) for i in model.predict(X_test)]

        con_mat = tf.math.confusion_matrix(labels=y_test, predictions=y_pred).numpy()

        con_mat_norm = np.around(con_mat.astype('float') / con_mat.sum(axis=1)[:, np.newaxis], decimals=2)
        label_names = list(range(len(con_mat_norm)))

        con_mat_df = pd.DataFrame(con_mat_norm,
                                  index=label_names,
                                  columns=label_names)

        figure = plt.figure(figsize=(10, 10))
        sns.heatmap(con_mat_df, cmap=plt.cm.Blues, annot=True)
        plt.ylabel('True label')
        plt.xlabel('Predicted label')
```

```
[ ] scores = model.evaluate(X_test, y_test, verbose=0)
    print("Accuracy: %.2f%%" % (scores[1] * 100))
```

```
WARNING:tensorflow:The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when calling a model.They have to be set to True/False
WARNING:tensorflow:The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
Accuracy: 88.25%
```

Figure 19: Evaluation of RoBERTa with aspect terms

The below figure is the confusion matrix on which x-axis is the predicted label and on the y-axis is the Actual label. From the figure it can be seen that the value was negative and it predicted negative for 86% and the value was positive and it was predicted positive 90% of the time.

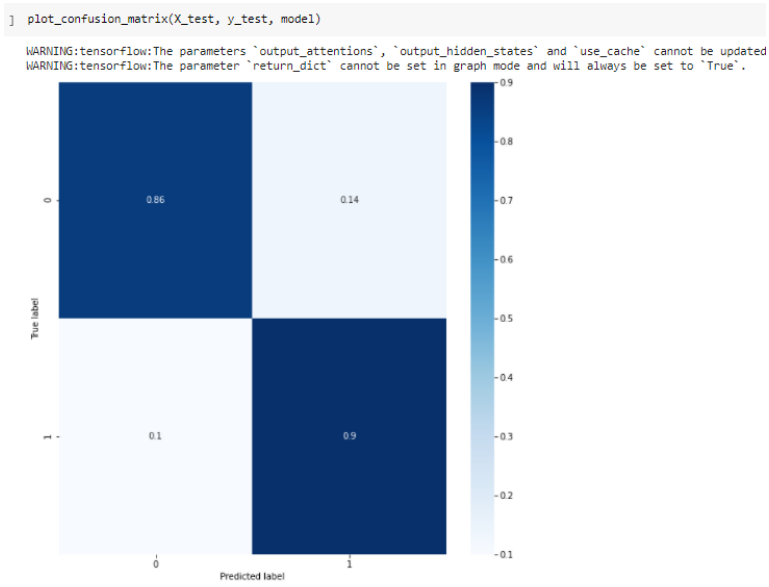

Figure 20: Plotting Confusion matrix for RoBERTa with aspect terms

3.4.4 ALBERT

In this research by Wang et al. (2020) a model of ALBERT and LSTM is proposed for sentiment analysis of product review. Here, first the pre-trained ALBERT model is used to get the word vector which contains positional and semantic data and LSTM model is used to obtain semantic features for training

```
class AlbertClassfier(torch.nn.Module):
    def __init__(self,bert_model,bert_config,num_class):
        super(AlbertClassfier,self).__init__()
        self.bert_model=bert_model
        self.dropout=torch.nn.Dropout(0.4)
        self.fc1=torch.nn.Linear(bert_config.hidden_size,bert_config.hidden_size)
        self.fc2=torch.nn.Linear(bert_config.hidden_size,num_class)
    def forward(self,token_ids):
        bert_out=self.bert_model(token_ids)[1] #Sentence vector [batch_size,hidden_size]
        bert_out=self.dropout(bert_out)
        bert_out=self.fc1(bert_out)
        bert_out=self.dropout(bert_out)
        bert_out=self.fc2(bert_out) #[batch_size,num_class]
        return bert_out
```

```
albertBertClassifier=AlbertClassfier(model,config,2)
device=torch.device("cuda:0") if torch.cuda.is_available() else 'cpu'
albertBertClassifier=albertBertClassifier.to(device)
```

Figure 21: building ALBERT model network structure

```
data=[]
label=[]

for index, row in tqdm(df.iterrows()):
    l = row["aspectAverageScore"]
    row=row["review_body"]
    ids=tokenizer.encode(row.strip(),max_length=100,padding='max_length',truncation=True)
    data.append(ids)
    label.append(l)

X_train, X_test, y_train, y_test=train_test_split(data,label,test_size=0.4,shuffle=True)
```

```
40000it [01:44, 383.94it/s]
```

```
from torch.utils import data
```

```
class DataGen(data.Dataset):
    def __init__(self,data,label):
        self.data=data
        self.label=label
    def __len__(self):
        return len(self.data)

    def __getitem__(self,index):
        return np.array(self.data[index]),np.array(self.label[index])
```

```
train_dataset=DataGen(X_train,y_train)
test_dataset=DataGen(X_test,y_test)
train_dataloader=data.DataLoader(train_dataset,batch_size=256)
test_dataloader=data.DataLoader(test_dataset,batch_size=256)
```

Figure 22: Preparation of training and testing data for ALBERT

The training and testing accuracy for the ALBERT model without aspect terms 83.06% and 83.80% respectively

```
for epoch in range(5):
    loss_sum=0.0
    accu=0
    albertBertClassifier.train()
    for step,(token_ids,label) in enumerate(train_dataloader):
        token_ids=token_ids.to(device)
        label=label.to(device)
        out=albertBertClassifier(token_ids)
        loss=criterion(out,label)
        optimizer.zero_grad()
        loss.backward() #Back propagation
        optimizer.step() #Gradient update
        loss_sum+=loss.cpu().data.numpy()
        accu+=(out.argmax(1)==label).sum().cpu().data.numpy()

    test_loss_sum=0.0
    test_accu=0
    albertBertClassifier.eval()
    for step,(token_ids,label) in enumerate(test_dataloader):
        token_ids=token_ids.to(device)
        label=label.to(device)
        with torch.no_grad():
            out=albertBertClassifier(token_ids)
            loss=criterion(out,label)
            test_loss_sum+=loss.cpu().data.numpy()
            test_accu+=(out.argmax(1)==label).sum().cpu().data.numpy()
    print("epoch % d,train loss:%f,train acc:%f,test loss:%f,test acc:%f"%(epoch,loss_sum/len(train_dataset),accu/len(train_dataset),test_loss_sum/len(test_dataset),test_accu/len(test_dataset)))
```

```
epoch  0,train loss:0.001839,train acc:0.825750,test loss:0.001746,test acc:0.838083
epoch  1,train loss:0.001800,train acc:0.830643,test loss:0.001748,test acc:0.838083
epoch  2,train loss:0.001794,train acc:0.830643,test loss:0.001748,test acc:0.838083
epoch  3,train loss:0.001794,train acc:0.830643,test loss:0.001745,test acc:0.838083
epoch  4,train loss:0.001790,train acc:0.830643,test loss:0.001737,test acc:0.838083
```

Figure 23: Evaluation of ALBERT model without aspect terms

The training and testing accuracy for the ALBERT model without aspect terms 59.07% and 59.06% respectively

```
] for epoch in range(5):
      loss_sum=0.0
      accu=0
      albertBertClassifier.train()
      for step,(token_ids,label) in enumerate(train_dataloader):
          token_ids=token_ids.to(device)
          label=label.to(device)
          out=albertBertClassifier(token_ids)
          loss=criterion(out,label)
          optimizer.zero_grad()
          loss.backward() #Back propagation
          optimizer.step() #Gradient update
          loss_sum+=loss.cpu().data.numpy()
          accu+=(out.argmax(1)==label).sum().cpu().data.numpy()

      test_loss_sum=0.0
      test_accu=0
      albertBertClassifier.eval()
      for step,(token_ids,label) in enumerate(test_dataloader):
          token_ids=token_ids.to(device)
          label=label.to(device)
          with torch.no_grad():
              out=albertBertClassifier(token_ids)
              loss=criterion(out,label)
              test_loss_sum+=loss.cpu().data.numpy()
              test_accu+=(out.argmax(1)==label).sum().cpu().data.numpy()
      print("epoch % d,train loss:%f,train acc:%f,test loss:%f,test acc:%f"%(epoch,loss_sum/len(train_dataset),accu/len(train_dataset),

epoch  0,train loss:0.002684,train acc:0.573583,test loss:0.002671,test acc:0.597625
epoch  1,train loss:0.002661,train acc:0.590750,test loss:0.002662,test acc:0.597625
```

Figure 24: Evaluating ALBERT model with aspect terms

References

Alshahrani, A., Ghaffari, M., Amirizirtol, K. and Liu, X. (2020). Identifying optimism and pessimism in twitter messages using xlnet and deep consensus, pp. 1–8.

Ghasiya, P. and Okamura, K. (2021). Investigating covid-19 news across four nations: A topic modeling and sentiment analysis approach, *IEEE Access* **9**: 36645–36656.

Wang, H., Hu, X. and Zhang, H. (2020). Sentiment analysis of commodity reviews based on ALBERT-LSTM, *Journal of Physics Conference Series*, Vol. 1651 of *Journal of Physics Conference Series*, p. 012022.