National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Jinal Sarvaiya
Student ID: x19207662

School of Computing
National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Jinal Sarvaiya……

Student ID: …x19207662……

Programme: …Msc in Data Analytics…… **Year:** 2021-2022…..

Module: …Msc in Research Project……

Lecturer: ……

Submission Due Date: ……31 January 2022……

Project Title: ”Multilingual Text Analysis using Transfer Learning and Natural Language Processing“…

Word Count: …993…… **Page Count:** ……16……

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: ……*Jinal Sarvaiya*……

Date: …31 January 2022……

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):	

Configuration Manual

Jinal Sarvaiya
x19207662

Introduction

The Configuration handbook lists the many parameters and configurations that were used to conduct this study, such as installation and requirements. This handbook contains a step-by-step explanation of how to run the application.

1 Configuration and Specification

1.1 Configuration (Hardware)

Hardware configuration screenshot of system details in Figure 1 and Figure 2 can be seen.

Figure 1

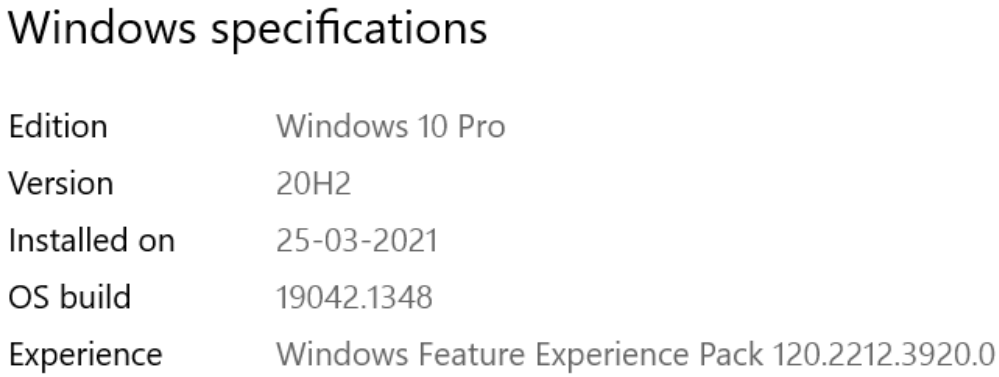

Figure 2

1.2 Configuration (Software)

Anaconda has been installed along with IDE such as Jupyter notebook.
Installation steps are as follows-

1.2.1 Anaconda installation

Steps 1: Figure 3 will be visible after we go to website: Anaconda.com/downloads

Figure 3

Steps 2: If it is a windows machine, based on your operating system then click windows. Windows was chosen in the proposed research shown in below figure 4.

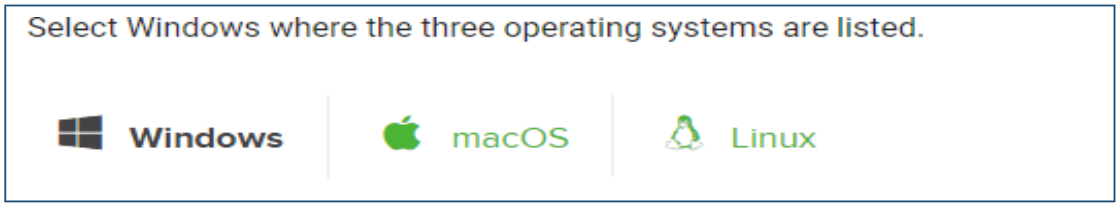

Figure 4

Steps 3: Python's latest Version 3.6 is installed.

Figure 5

Step 4: Installation of exe file was done after downloading and follow steps in the images shown below:

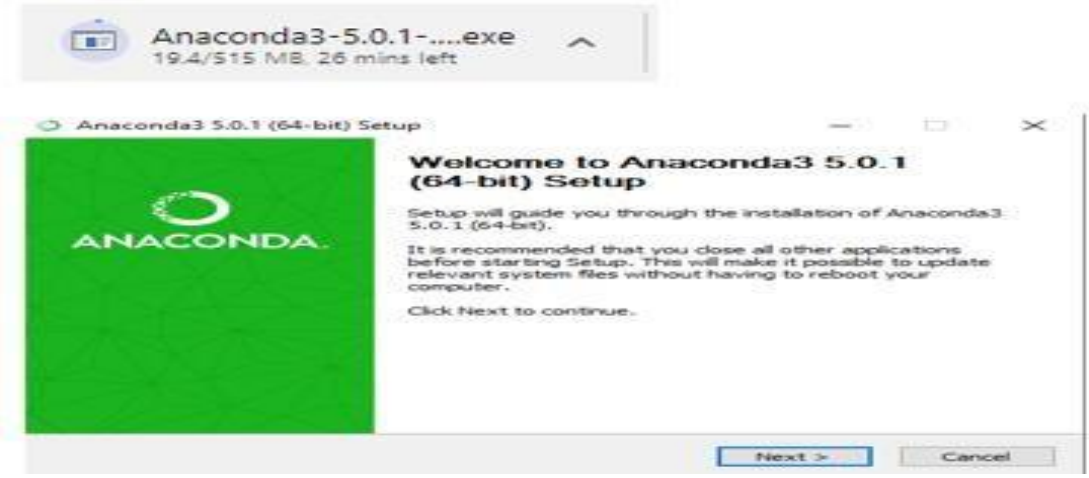

Figure 6

Step 5: Click on "I Agree" and click next which starts with installing the software. Figure 7.

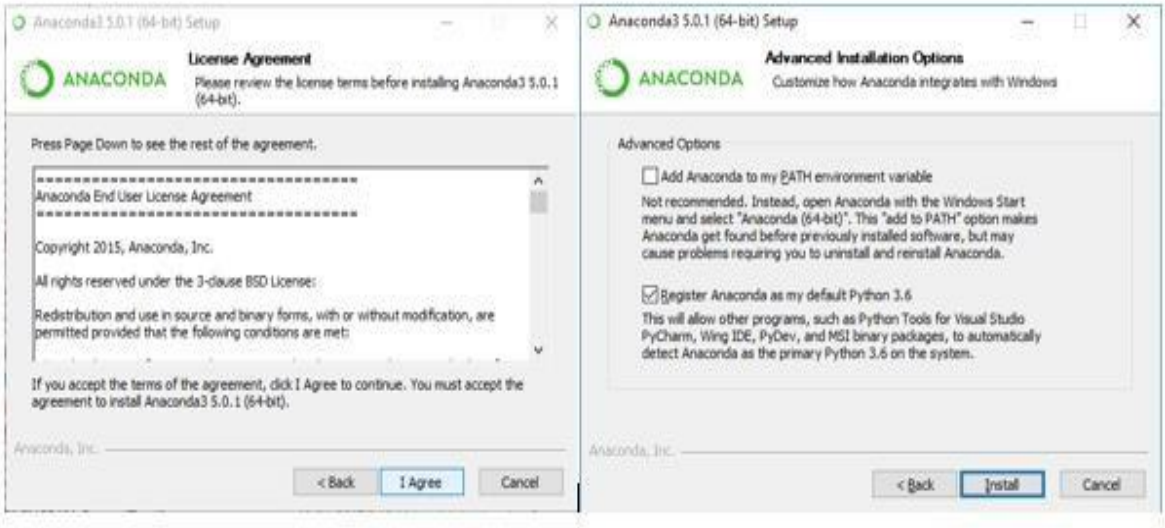

Figure 7

Step 6: Click on anaconda to open

1.2.2 Installation of Jupyter

Step 1: After installation, from the application launch "Jupyter". There are multiple options of IDE which are visible in Navigator. The version we require is on use case any IDE we use.
Notebook shown in figure **8.**

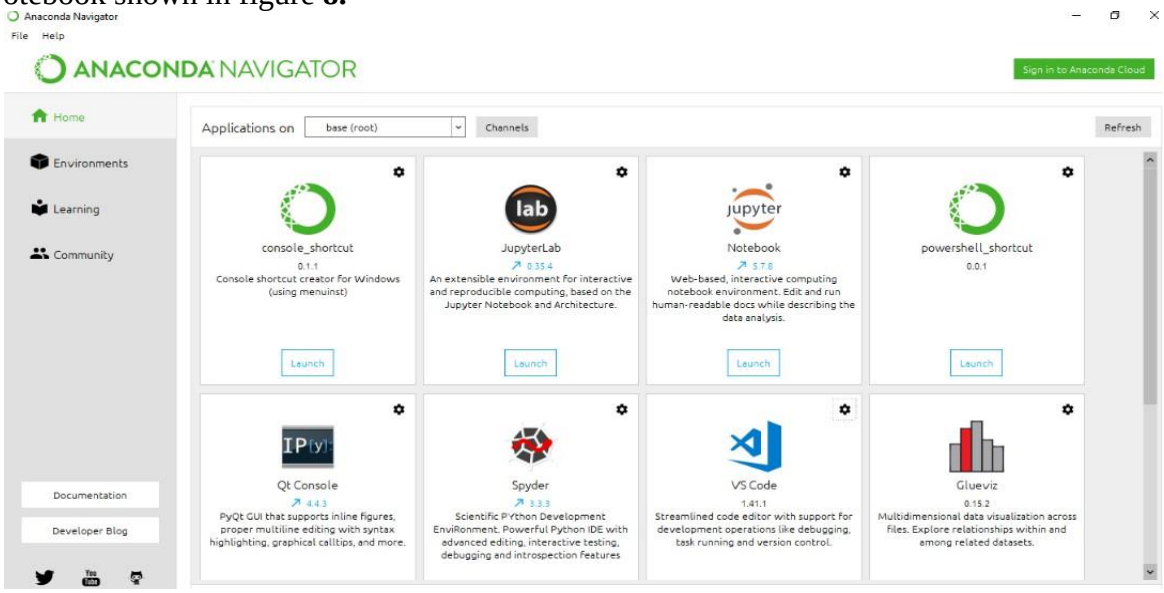

Figure 8

1.2.3 Opening Colab:

The model is trained using Google Colab. Colab, like a Jupyter notebook. By selecting a GPU or TPU, one can adjust the runtime

Step 1: We open google drive > app > right click> More > google colab
Below is the page which will be visible and rename the file

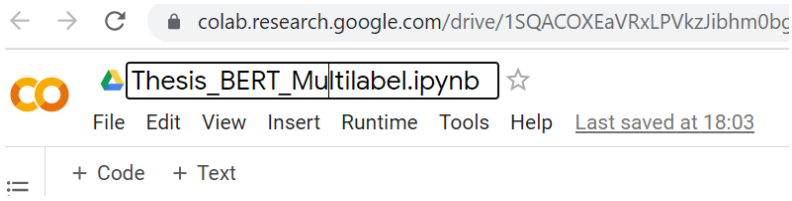

Figure 9

Step 2: "Runtime" > runtime change > select GPU > SaveRefer Figure 10.

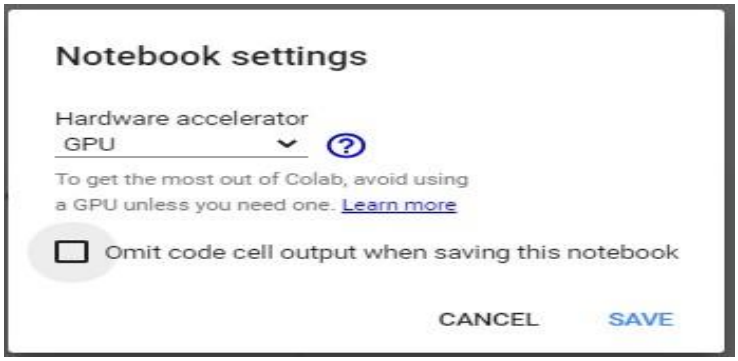

Figure 10

1.3 Importing libraries required for data cleaning .

```
import requests
from nltk.corpus import stopwords
from nltk import FreqDist
import seaborn as sns
import os, json
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
import re,string,unicodedata
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
from string import punctuation
from nltk import pos_tag
from nltk.corpus import wordnet
from sklearn.preprocessing import OneHotEncoder
nltk.download('wordnet')
from googletrans import Translator
import googletrans
import requests
from nltk.corpus import stopwords
from nltk import FreqDist
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelBinarizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.multiclass import OneVsRestClassifier
```

Code 1

1.4 Concatenating all uncleaned CSVs

Code 2 shows code for concatenation of all datasets into one CSV (ds_es)

```
df_sp = pd.read_csv(r'jigsaw-toxic-comment-train-google-es.csv')
df_fr = pd.read_csv(r'jigsaw-toxic-comment-train-google-fr.csv')
df_it = pd.read_csv(r'jigsaw-toxic-comment-train-google-it.csv')
```

```
df_es = pd.concat(map(pd.read_csv, ['jigsaw-toxic-comment-train-google-es.csv', 'jigsaw-toxic-comment-train-google-fr.csv', '
```

Code 2

1.5 Pre-processing steps

Refer Code 3 for preprocessing steps by removing null values and unwanted columns

```
df_es.pop('Unnamed: 0' )
df_es.pop('Unnamed: 0.1')
df_es.pop('Unnamed: 0.1.1')
```

```
0             20
1             13
2              0
3              6
4             25
           ...
159368    159565
159369    159555
159370    159586
159371    159571
159372    159560
Name: Unnamed: 0.1.1, Length: 159373, dtype: object
```

```
df_es.isna().sum()
```

```
id               0
cleaned_text     0
toxic            0
severe_toxic     0
obscene          0
threat           0
insult           0
identity_hate    0
comment_text     0
dtype: int64
```

Code 3

1.6 Use of Google Translator API

Translation of Google API from Spanish, Italian and French to English

```
from googletrans import Translator

translator = Translator()
translations = []
for element in df_es['comment_text']:
    translations.append(translator.translate(element).text)
df_es['translations'] = translations
```

Code 4

1.7 POS Tagging and Lemmatization

```
def posTagger(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

```
#Lemmitization

wordLemmatize = WordNetLemmatizer()
def applyLemmatizer(text):
    wordVec = []
    for i in text.split():
        if i.strip().lower() not in stopwordsTxt:
            temp = pos_tag([i.strip()])
            word = wordLemmatize.lemmatize(i.strip(),posTagger(temp[0][1]))
            wordVec.append(word.lower())
    return " ".join(wordVec)
```

Code 5

1.8 Label Assignment

Labels (toxic, severe_toxic, identity_hate, obscene, threat, insult) has been assigned to column "tag"

```
df_es_label= df_es[[ 'toxic', 'severe_toxic','obscene', 'threat', 'insult', 'identity_hate']]
```

```
df_es.columns[2:8]
```

```
Index(['toxic', 'severe_toxic', 'obscene', 'threat', 'insult',
       'identity_hate'],
      dtype='object')
```

```
df_es_label['tags'] = ''

for col_name in list(df_es_label.columns):
    df_es_label.loc[df_es_label[col_name] == '1', 'tags'] = df_es_label['tags'] + " " + col_name

print(df_es_label)
```

Code 6

	toxic	severe_toxic	obscene	threat	insult	identity_hate	tags
0	0	0	0	0	0	0	
1	0	0	0	0	0	0	
2	0	0	0	0	0	0	
3	0	0	0	0	0	0	
4	0	0	0	0	0	0	
5	0	0	0	0	0	0	
6	1	1	1	0	1	0	toxic severe_toxic obscene insult
7	0	0	0	0	0	0	
8	0	0	0	0	0	0	
9	0	0	0	0	0	0	
10	0	0	0	0	0	0	
11	0	0	0	0	0	0	
12	1	0	0	0	0	0	toxic

Code 7

1.9 Outputs of Traditional Models for MNB, SVC, Logistic Regression

```
Test accuracy is 0.9175


Test F1 score is 0.11959003038921535


Classification Report:
                                                precision    recall  f1-score   support

                                                     0.92      1.00      0.96      1817
                                            insult       0.00      0.00      0.00         3
                                insult identity_hate       0.00      0.00      0.00         1
                                           obscene       0.00      0.00      0.00         5
                                     obscene insult       0.00      0.00      0.00         5
                           obscene insult identity_hate       0.00      0.00      0.00         1
                                             toxic       0.00      0.00      0.00        58
                                toxic identity_hate       0.00      0.00      0.00         1
                                       toxic insult       1.00      0.22      0.36         9
                          toxic insult identity_hate       1.00      0.50      0.67         2
                                       toxic obscene       0.00      0.00      0.00        16
                                toxic obscene insult       0.54      0.29      0.38        48
                       toxic obscene insult identity_hate       0.00      0.00      0.00        12
                              toxic obscene threat insult       0.00      0.00      0.00         1
                toxic obscene threat insult identity_hate       0.00      0.00      0.00         2
                                   toxic severe_toxic       0.00      0.00      0.00         1
                            toxic severe_toxic obscene       0.00      0.00      0.00         1
                     toxic severe_toxic obscene insult       1.00      0.08      0.14        13
         toxic severe_toxic obscene insult identity_hate       0.00      0.00      0.00         1
                toxic severe_toxic obscene threat insult       0.00      0.00      0.00         1
                                       toxic threat       0.00      0.00      0.00         2

                                           accuracy                           0.92      2000
                                          macro avg       0.21      0.10      0.12      2000
                                       weighted avg       0.86      0.92      0.88      2000
```

```
Confusion Matrix:
[[28264     2     1 ...     0     0     0]
 [    8     0     0 ...     0     0     0]
 [   48     0     0 ...     0     0     0]
 ...
 [    1     0     0 ...     0     0     0]
 [    0     0     0 ...     0     0     0]
 [    1     0     0 ...     0     0     0]]
```

Code 8- Confusion Matrix for Logistic Regression

	precision	recall	f1-score	support
	0.96	0.99	0.97	28574
identity_hate	0.00	0.00	0.00	11
insult	0.00	0.00	0.00	65
insult identity_hate	0.00	0.00	0.00	7
obscene	0.24	0.08	0.12	63
obscene insult	0.00	0.00	0.00	43
obscene insult identity_hate	0.00	0.00	0.00	5
threat	0.00	0.00	0.00	3
threat insult	0.00	0.00	0.00	1
toxic	0.29	0.22	0.25	1129
toxic identity_hate	0.00	0.00	0.00	30
toxic insult	0.27	0.11	0.16	250
toxic insult identity_hate	0.00	0.00	0.00	32
toxic obscene	0.31	0.19	0.24	361
toxic obscene identity_hate	0.00	0.00	0.00	7
toxic obscene insult	0.44	0.54	0.49	765
toxic obscene insult identity_hate	0.23	0.11	0.15	135
toxic obscene threat	0.00	0.00	0.00	1
toxic obscene threat insult	0.17	0.05	0.08	20
toxic obscene threat insult identity_hate	0.12	0.17	0.14	6
toxic severe_toxic	0.00	0.00	0.00	9
toxic severe_toxic insult	0.00	0.00	0.00	2
toxic severe_toxic insult identity_hate	0.00	0.00	0.00	4
toxic severe_toxic obscene	0.12	0.03	0.04	38
toxic severe_toxic obscene identity_hate	0.00	0.00	0.00	2
toxic severe_toxic obscene insult	0.35	0.21	0.26	201
toxic severe_toxic obscene insult identity_hate	0.33	0.04	0.07	55
toxic severe_toxic obscene threat	0.00	0.00	0.00	1
toxic severe_toxic obscene threat insult	0.00	0.00	0.00	15
toxic severe_toxic obscene threat insult identity_hate	0.00	0.00	0.00	5
toxic severe_toxic threat	0.25	1.00	0.40	1
toxic threat	0.12	0.03	0.05	31
toxic threat identity_hate	0.00	0.00	0.00	1
toxic threat insult	0.00	0.00	0.00	1
toxic threat insult identity_hate	0.00	0.00	0.00	1
accuracy			0.91	31875
macro avg	0.12	0.11	0.10	31875
weighted avg	0.89	0.91	0.90	31875

```
Confusion Matrix:
[[28264     2     1 ...     0     0     0]
 [    8     0     0 ...     0     0     0]
 [   48     0     0 ...     0     0     0]
 ...
 [    1     0     0 ...     0     0     0]
 [    0     0     0 ...     0     0     0]
 [    1     0     0 ...     0     0     0]]
```

Code 9- Confusion matrix of MNB

```
Test accuracy is 0.9175


Test F1 score is 0.11959003038921535


Classification Report:
                                                precision    recall  f1-score   support

                                                     0.92      1.00      0.96      1817
                                            insult       0.00      0.00      0.00         3
                                insult identity_hate       0.00      0.00      0.00         1
                                             obscene       0.00      0.00      0.00         5
                                      obscene insult       0.00      0.00      0.00         5
                          obscene insult identity_hate       0.00      0.00      0.00         1
                                               toxic       0.00      0.00      0.00        58
                                toxic identity_hate       0.00      0.00      0.00         1
                                        toxic insult       1.00      0.22      0.36         9
                          toxic insult identity_hate       1.00      0.50      0.67         2
                                      toxic obscene       0.00      0.00      0.00        16
                                toxic obscene insult       0.54      0.29      0.38        48
                    toxic obscene insult identity_hate       0.00      0.00      0.00        12
                           toxic obscene threat insult       0.00      0.00      0.00         1
              toxic obscene threat insult identity_hate       0.00      0.00      0.00         2
                                 toxic severe_toxic       0.00      0.00      0.00         1
                         toxic severe_toxic obscene       0.00      0.00      0.00         1
                   toxic severe_toxic obscene insult       1.00      0.08      0.14        13
      toxic severe_toxic obscene insult identity_hate       0.00      0.00      0.00         1
             toxic severe_toxic obscene threat insult       0.00      0.00      0.00         1
                                       toxic threat       0.00      0.00      0.00         2

                                           accuracy                           0.92      2000
                                          macro avg       0.21      0.10      0.12      2000
                                       weighted avg       0.86      0.92      0.88      2000
```

Code 10- Output for SVC

2 Download Dataset

The dataset is downloaded from “Kaggle repository”, from link https://www.kaggle.com/miklgr500/jigsaw-train-multilingual-comentsgoogle-api (1)

Alternative way, Steps to download the cropped dataset and upload to Google drive.

1. Download the dataset [Dataset] from the following OneDrive link : https://studentncirl-my.sharepoint.com/:f:/g/personal/x19207662_student_ncirl_ie/Ely8NW0cYuZAkY8x8JqWkbEBsWhZEnnsMgXisERH_NyPWQ?e=jPQnL1
2. Upload the folder [Dataset] into Google Drive.

3 Prepare setup for data modelling by BERT

3.1 Uploading the corpus to google drive

Under folder name “Thesis” , we can upload complete dataset containing BERT on drive in “Thesis folder”. figure 11 is visual presentation of it.

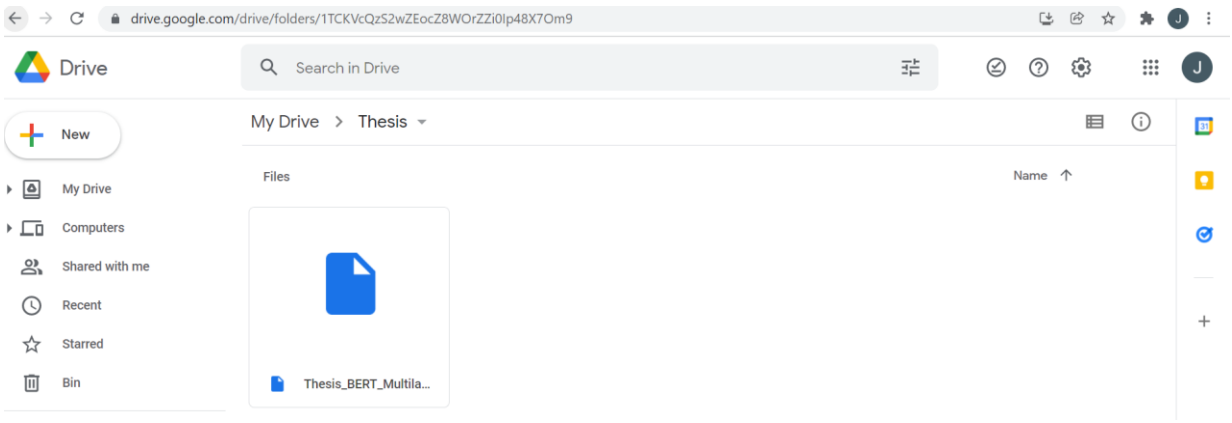

Figure 11

3.2 Drive to Collaboratory notebook connection

Data to collaboratory notebook connection code is shown in Code 11.

```
from google.colab import drive
drive.mount ('/content/drive')
```

Code 11

There is a authorization code which is obtained after connection to google drive. The previous tab will be opened in new window and after copying the key, paste in code in code 11

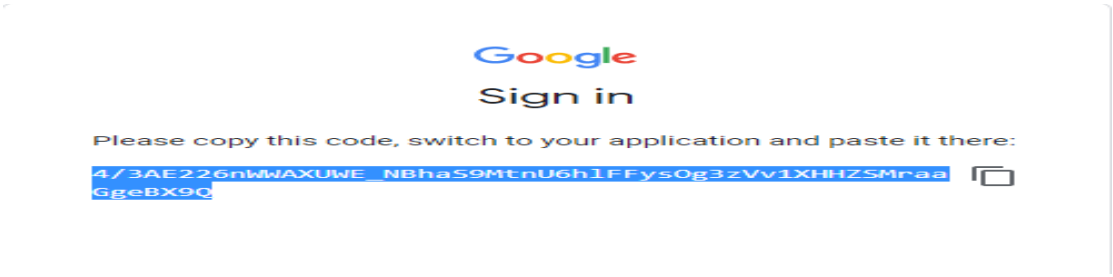

Figure 12

3.3 Importing Transformers

```
# Installing the transformers library and additional libraries if looking process

!pip install -q transformers==3

# Code for TPU packages install
# !curl -q https://raw.githubusercontent.com/pytorch/xla/master/contrib/scripts/env-setup.py -o pytorch-xla-env-setup.py
# !python pytorch-xla-env-setup.py --apt-packages libomp5 libopenblas-dev
```

```
     |████████████████████████████████| 754 kB 5.6 MB/s
     |████████████████████████████████| 895 kB 32.1 MB/s
     |████████████████████████████████| 1.2 MB 34.8 MB/s
     |████████████████████████████████| 3.0 MB 32.6 MB/s
```

Code 12

3.4 Import libraries of py which set the root directories.

```
import numpy as np
import pandas as pd
from sklearn import metrics
import transformers
import torch
from torch.utils.data import Dataset, DataLoader, RandomSampler, SequentialSampler
from transformers import BertTokenizer, BertModel, BertConfig

import requests
from nltk.corpus import stopwords
from nltk import FreqDist
import seaborn as sns
import os, json
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
import re,string,unicodedata
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
from sklearn.linear_model import LogisticRegression,SGDClassifier
```

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
from string import punctuation
from nltk import pos_tag
from nltk.corpus import wordnet
from sklearn.preprocessing import OneHotEncoder
nltk.download('wordnet')
#from googletrans import Translator
#import googletrans
import requests
from nltk.corpus import stopwords
from nltk import FreqDist
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelBinarizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
nltk.download('averaged_perceptron_tagger')
```

Code 13

3.4 Create Main "Config" function

```
# Sections of config
# Defining some key variables that will be used later on in the training

MAX_LEN = 350
TRAIN_BATCH_SIZE = 20
VALID_BATCH_SIZE = 5
EPOCHS = 7
LEARNING_RATE = 1e-05
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

Code 14

3.5 Load Data with BERT file and add class

```
class CustomDataset(Dataset):

    def __init__(self, dataframe, tokenizer, max_len):
        self.tokenizer = tokenizer
        self.data = dataframe
        self.comment_text = dataframe.comment_text
        self.targets = self.data.list
        self.max_len = max_len

    def __len__(self):
        return len(self.comment_text)

    def __getitem__(self, index):
        comment_text = str(self.comment_text[index])
        comment_text = " ".join(comment_text.split())

        inputs = self.tokenizer.encode_plus(
            comment_text,
            None,
            add_special_tokens=True,
            max_length=self.max_len,
            pad_to_max_length=True,
            truncation=True,
            return_token_type_ids=True
        )
        ids = inputs['input_ids']
        mask = inputs['attention_mask']
        token_type_ids = inputs["token_type_ids"]


        return {
            'ids': torch.tensor(ids, dtype=torch.long),
            'mask': torch.tensor(mask, dtype=torch.long),
            'token_type_ids': torch.tensor(token_type_ids, dtype=torch.long),
            'targets': torch.tensor(self.targets[index], dtype=torch.float)
        }
```

Code 15

3.6 Split into train and Validation

Train and Validation code can be seen in code 16.

```
# Creating the dataset and dataloader for the neural network

train_size = 0.7
train_dataset=new_df.sample(frac=train_size,random_state=200)
test_dataset=new_df.drop(train_dataset.index).reset_index(drop=True)
train_dataset = train_dataset.reset_index(drop=True)


print("FULL Dataset: {}".format(new_df.shape))
print("TRAIN Dataset: {}".format(train_dataset.shape))
print("TEST Dataset: {}".format(test_dataset.shape))

training_set = CustomDataset(train_dataset, tokenizer, MAX_LEN)
testing_set = CustomDataset(test_dataset, tokenizer, MAX_LEN)
```

```
FULL Dataset: (100, 2)
TRAIN Dataset: (70, 2)
TEST Dataset: (30, 2)
```

```
train_params = {'batch_size': TRAIN_BATCH_SIZE,
                'shuffle': True,
                'num_workers': 0
                }

test_params = {'batch_size': VALID_BATCH_SIZE,
                'shuffle': True,
                'num_workers': 0
                }

training_loader = DataLoader(training_set, **train_params)
testing_loader = DataLoader(testing_set, **test_params)
```

Code 16

3.7 Slip dataset into train and Validation

The code 17 shows, model is created in training mode.

```
# Creating the customized model, by adding a drop out and a dense layer on top of distil bert to get the final output for the model.

class BERTClass(torch.nn.Module):

    def __init__(self):
        super(BERTClass, self).__init__()
        self.l1 = transformers.BertModel.from_pretrained('bert-base-uncased')
        self.l2 = torch.nn.Dropout(0.3)
        self.l3 = torch.nn.Linear(768, 6)

    def forward(self, ids, mask, token_type_ids):
        _, output_1= self.l1(ids, attention_mask = mask, token_type_ids = token_type_ids)
        output_2 = self.l2(output_1)
        output = self.l3(output_2)
        return output

model = BERTClass()

model.to(device)
```

Code 17

3.8 Train model for 5 epochs

```
def train(epoch):
    model.train()
    for _,data in enumerate(training_loader, 0):
        ids = data['ids'].to(device, dtype = torch.long)
        mask = data['mask'].to(device, dtype = torch.long)
        token_type_ids = data['token_type_ids'].to(device, dtype = torch.long)
        targets = data['targets'].to(device, dtype = torch.float)

        outputs = model(ids, mask, token_type_ids)

        optimizer.zero_grad()
        loss = loss_fn(outputs, targets)
        if _%5000==0:
            print(f'Epoch: {epoch}, Loss:  {loss.item()}')

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

```
for epoch in range(0, EPOCHS):
  train(epoch)
```

```
Epoch: 0, Loss:  0.7326139211654663
Epoch: 1, Loss:  0.5491685271263123
Epoch: 2, Loss:  0.4489758610725403
Epoch: 3, Loss:  0.41286396980285645
Epoch: 4, Loss:  0.34120652079582214
```

Code 18

References

Kaggle repository. (2019). https://www.kaggle.com/miklgr500/jigsaw-train-multilingual-coments-google-api, CC0 1.0 Universal (CC0 1.0) Public Domain Dedication