National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Priyanka Gupta
Student ID: x19223030

School of Computing
National College of Ireland

Supervisor: Bharathi Chakravarthi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Priyanka Gupta……

Student ID: X19223030.……

Programme: Data Analytics…… **Year:** 2021…….

Module: MSc Research Project……

Lecturer: Bharathi Chakravarthi……

Submission Due Date: 16-08-2021……

Project Title: Leveraging Transfer learning techniques- BERT, RoBERTa, ALBERT and DistilBERT for Fake Review Detection

Page Count: 13
Word Count: 565

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: 16-08-2021……

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):	

Configuration Manual

Priyanka Gupta
X19223030

1 Introduction

This configuration manual describes the hardware and the software used to build fake review detection system. The steps mentioned in this manual can be followed to reproduce our results and build a fake review classifier using transfer learning models.

2 Hardware and Software Requirement

To train the transfer learning-based models, BERT, RoBERTa, DistilBERT, and ALBERT, Google Collaboratory cloud machine is used due to large training data. The specification of the host device is given below.

Inspiron 7501

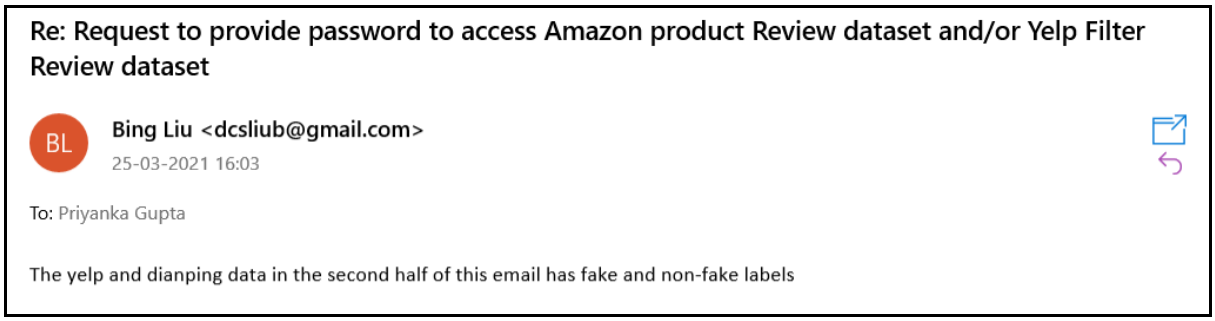
Device name	DESKTOP-MVPSC50
Processor	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
Installed RAM	16.0 GB (15.8 GB usable)
Device ID	16D07654-321F-4330-ABC9-538AEAFF7327
Product ID	00327-35898-54617-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

The project is implemented using Python 3.7, and the required libraries are mention in the below table. A Gmail account is mandatory to access Google Collaboratory. The implementation of our project is carried out on Google Colab Pro. Also, the dataset is available in .db extension, hence SQLite Database is required.

IDE	Google Colab Pro (Cloud-based)
Computation	GPU
Type	Tesla P100-PCIE-16GB
Number of GPU	1
Programming language	Python
Framework	Pytorch
Modeling library	SimpleTransformer, HuggingFace Transformer, Sklearn, Pandas, Numpy, Matplotlib, Seaborn, Wandb

3 Dataset

The data used in our research project is collected by Mukherjee et al. (2013). It can be obtained by contacting the researchers via email (dcsliub@gmail.com).



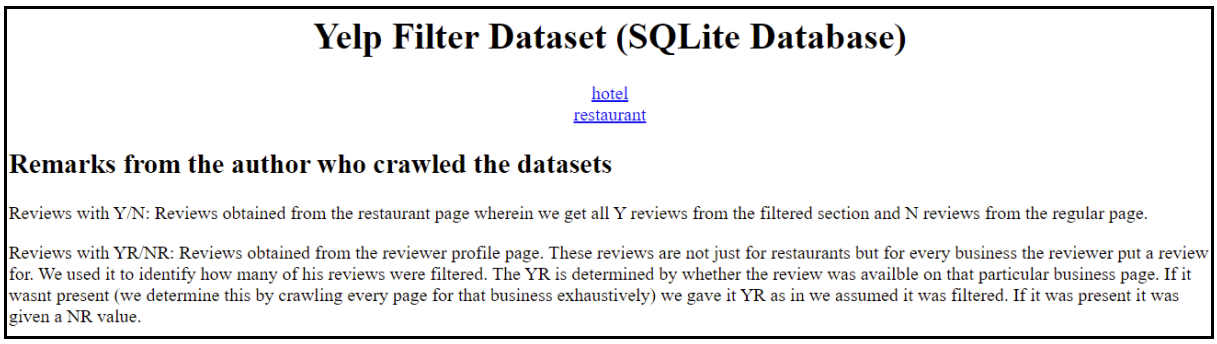
Re: Request to provide password to access Amazon product Review dataset and/or Yelp Filter Review dataset

BL Bing Liu <dcsliub@gmail.com>
25-03-2021 16:03

To: Priyanka Gupta

The yelp and dianping data in the second half of this email has fake and non-fake labels

The yelp data is provided with .db extension and can be opened using SQLite Database.



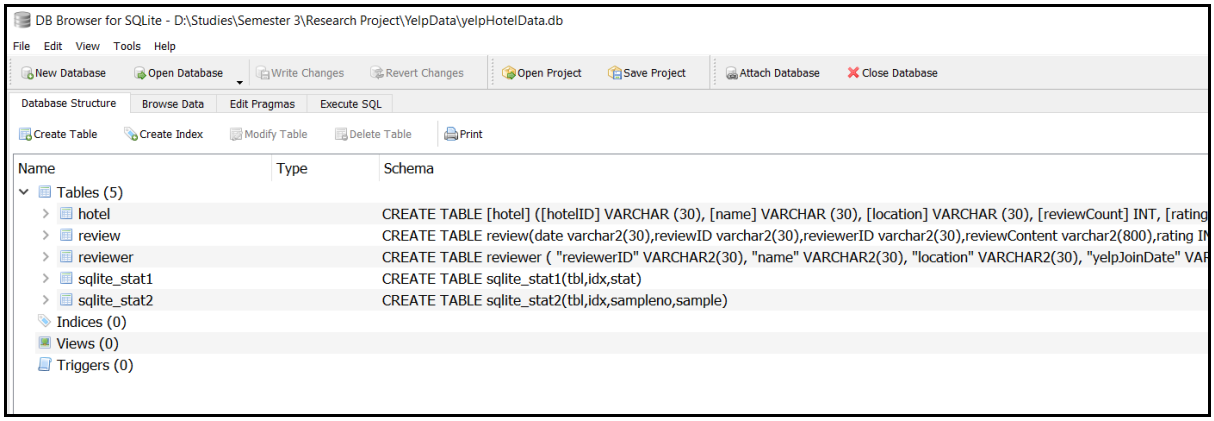
Yelp Filter Dataset (SQLite Database)

hotel
restaurant

Remarks from the author who crawled the datasets

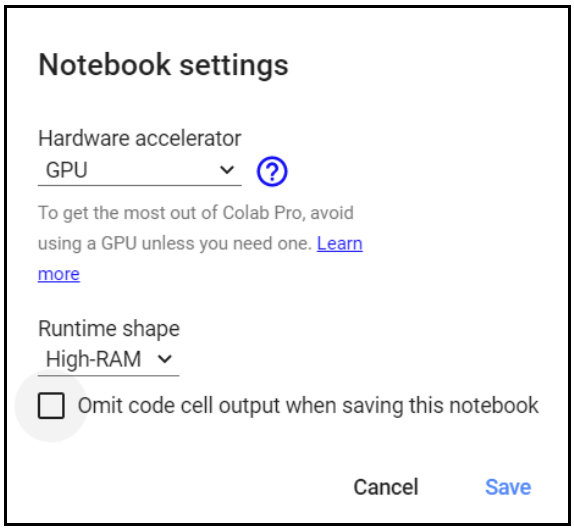
Reviews with Y/N: Reviews obtained from the restaurant page wherein we get all Y reviews from the filtered section and N reviews from the regular page.

Reviews with YR/NR: Reviews obtained from the reviewer profile page. These reviews are not just for restaurants but for every business the reviewer put a review for. We used it to identify how many of his reviews were filtered. The YR is determined by whether the review was availble on that particular business page. If it wasnt present (we determine this by crawling every page for that business exhaustively) we gave it YR as in we assumed it was filtered. If it was present it was given a NR value.

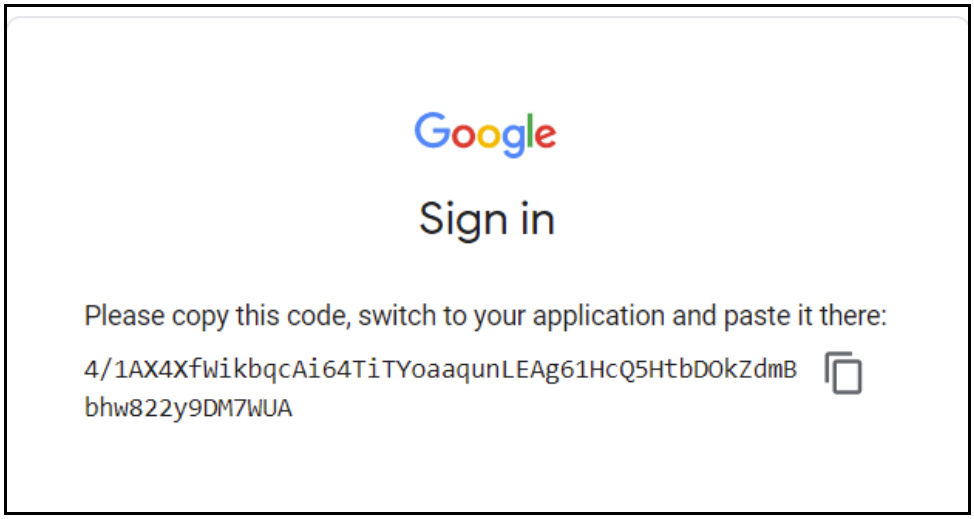


4 Implementation

Post logging into Google colab Pro, Notebook setting needs to be change to GPU and High-RAM.



The dataset (.db file) should be uploaded to Google Drive and using below code, the notebook is mounted to google drive. User needs to click on below link, login to Gmail and enter the authorization code in notebook.



Also, using the Gmail account, the user needs to create a Weights & Biases account by going on https://wandb.ai/site. This is used for visualizing model training.

Importing Libraries for Data loading, EDA and Data cleaning.

```
# Importing required library
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
from tqdm import tqdm
tqdm.pandas()
import matplotlib.pyplot as plt
!pip install contractions
import contractions
import re
```

Connecting with SQLite database to load first dataset to Python and storing in a Dataframe

```
[4]  # Connecting with Database to load the data
     import pandas as pd
     import sqlite3

     # Read sqlite query results into a pandas DataFrame
     con = sqlite3.connect("/content/drive/MyDrive/yelpHotelData.db")
     df1 = pd.read_sql_query("SELECT * from review", con)

     # Verify that result of SQL query is stored in the dataframe
     print(df1.head(1))

     con.close()
```

Loading second dataset from database to Python and storing in a Dataframe

```
# Read sqlite query results into a pandas DataFrame
con = sqlite3.connect("/content/drive/MyDrive/yelpResData.db")
con.text_factory = lambda b: b.decode(errors = 'ignore')
df2 = pd.read_sql_query("SELECT * from review", con)

# Verify that result of SQL query is stored in the dataframe
print(df2.head(1))

con.close()
```

Renaming few column names and merging two dataframe to consolidate the data

```
#Renaming columns prior to combining the dataframes
df1.rename(columns= {'hotelID':'hotel/restaurantID'},inplace= True)
df2.rename(columns= {'restaurantID':'hotel/restaurantID'},inplace= True)

#Combining dataframes
df=pd.concat([df1,df2])
df.reset_index(drop=True, inplace=True)
```

Marking fake reviews as 1 and genuine reviews as 1

```
# Making necessary changes in flagged column of DataFrame
df["flagged"]= df["flagged"].replace('YR', 'Y')
df["flagged"]= df["flagged"].replace('NR', 'N')

#Selecting required columns and converting fake and non-fake reviews to 0 and 1
df=df.loc[:,['reviewID','reviewContent','flagged']]
df["flagged"]= df["flagged"].replace('Y', 1)
df["flagged"]= df["flagged"].replace('N', 0)
```

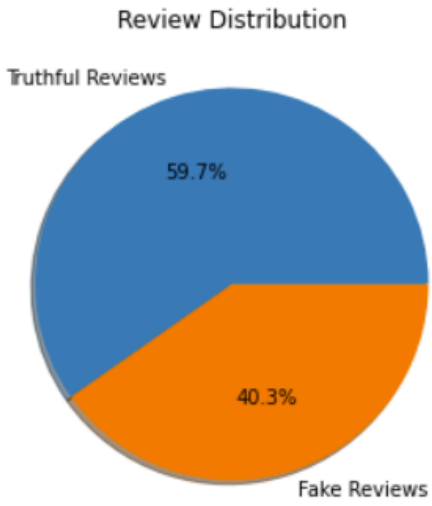
Checking for null values in dataset

```
df.isna().sum()

reviewID        0
reviewContent   0
flagged         0
dtype: int64
```

Plotting Pie chart to understand distribution of fake and truthful reviews

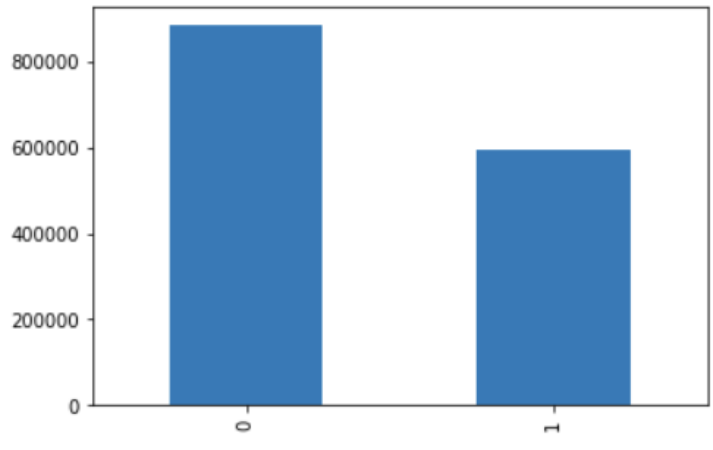
```
values = [df[df['flagged']== 0].shape[0], df[df['flagged']==1].shape[0]]
labels = ['Truthful Reviews', 'Fake Reviews']
plt.pie(values, labels=labels, autopct='%1.1f%%', shadow=True)
plt.title('Review Distribution')
plt.tight_layout()
plt.subplots_adjust(right=1.9)
plt.show()
```



Plotted bar graph to understand count of reviews

```
[19] df.flagged.value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fec3be8b250>
```



Defining and applying functions to expand contractions, remove puntuations, special characters and digits present in the reviews.

```
[22] # Defining function to remove contractions
     def clean_text(txt):
         cleaned_text= contractions.fix(txt)
         return cleaned_text

     #Applying function
     df['cleaned_reviews']  = df['reviewContent'].progress_apply(lambda txt: clean_text(txt))

     100%|██████████| 1476800/1476800 [01:26<00:00, 17005.74it/s]
```

```
[23] #Defining function to remove punctuations in reviews
     def remove_punch(text):
         res = re.sub(r'[^a-zA-Z]', ' ', text)
         return res
     # Apply function
     df['cleaned_reviews']  = df['cleaned_reviews'].progress_apply(lambda text: remove_punch(text))

     100%|██████████| 1476800/1476800 [00:51<00:00, 28544.13it/s]
```

```
[24] #Defining function to remove numeric digits in reviews
     def remove_num(text):
       # using filter and lambda
       # to remove numeric digits from string
       response = "".join(filter(lambda x: not x.isdigit(), text))
       return response

     #Applying function
     df['cleaned_reviews']  = df['cleaned_reviews'].progress_apply(lambda text: remove_num(text))
```

Checcking for duplicate reviews in the dataset and removing the same

```
# Checking the count of Duplicate reviews present
df['cleaned_reviews'].duplicated().sum()
```

```
190827
```

```
# Removing the duplicate reviews
df.drop_duplicates(subset ="cleaned_reviews",
                   keep = False, inplace = True)
```

```
# Verifying Duplicate reviews are removed and resetting the index
df['cleaned_reviews'].duplicated().sum()
df.reset_index(inplace = True)
```

Selecting columns required for model training and converting DataFrame to csv

```
# Selecting columns useful for further use
df=df.iloc[:, [4,3]]
```

```
# Validating the dataFrame
df.head(5)
```

	cleaned_reviews	flagged
0	The only place inside the Loop that you can st...	0
1	This place is disgusting absolutely horrible ...	0
2	This hotel came up on Hotwire for a night...	0
3	Good location really run down I am surprised...	0
4	Beautiful lobby The rest is a dump The eleva...	0

```
# Converting to CSV for further use
df.to_csv('Fake_Reviews_cleanedData.csv')
```

Pretrained Model Implenmentation:

Fine-tuning RoBERTa with fake review detection dataset

Installing Transformer and Simple Transformer library

```
# Installing Transformer and Simple Transformer library
!pip install -U transformers
!pip install -U simpletransformers
```

Importing other ClassificationModel from Simple transformers library along with other essential libraries.

```
#Importing other essential libraries
from simpletransformers.classification import ClassificationModel
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

Mounting Google drive and loading the cleaned data from CSV to Pandas DataFrame.

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

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

```
# Converting CSV to Pandas DataFrame
df=pd.read_csv("/content/drive/MyDrive/Fake_Reviews_cleanedData.csv")
```

```
# Selecting columns from dataDrame that are required for model implementation
df=df.iloc[:,1:]
```

Selecting 50% of cleaned dataset using df.sample and printing the shape. The value of frac can be set to 0.1 when 10% of data is required for model training.

```
#Data Sampling
#Selecting 50% data from the dataset
df=df.sample(frac=0.5)
```

```
# Printing the shape of 50% dataset
df.shape
```

```
(548502, 2)
```

Dividing the data into Train-Evaluation-Test sets

```
# Storing the user revies in X and corresponding labels in Y
X = df['cleaned_reviews']
y = df['flagged']
```

```
X.shape
```

```
(548502,)
```

```
# splitting the data into Test, Eval and train subsets
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.8)
```

```
test_size = 0.5
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.5)
```

Printing the shape of Train-Evaluation-Test sets

```
# Printing the shape of each subset
print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
print(X_test.shape), print(y_test.shape)
```

```
(438801,)
(438801,)
(54850,)
(54850,)
(54851,)
(54851,)
(None, None)
```

Converting the data to dataFrame

```
# Converting Train and Eval data into DataFrame
train_df = pd.DataFrame(X_train)
train_df['flagged'] = y_train

eval_df = pd.DataFrame(X_valid)
eval_df['flagged'] = y_valid

train_df.shape, eval_df.shape
```

```
((438801, 2), (54850, 2))
```

```
# Converting the testing reviews into DataFrame
test_df=pd.DataFrame(X_test)
```

```
# Printing test reviews
test_df.head()
```

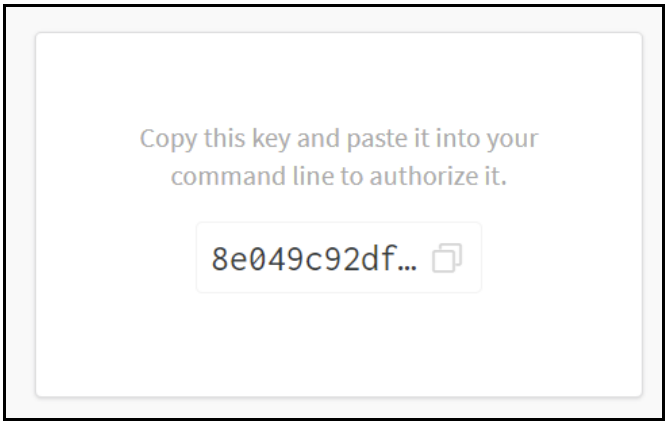
Creating a binary Classification Model for RoBERTa and providing hyper-parameters values. Also, train the model with training data and evaluating using eval data to prevent overfitting

```
results=[]
# Creating a ClassificationModel and providing selected hyper-parameters
model = ClassificationModel('roberta', 'roberta-base', args={
    'num_train_epochs':2,
    'train_batch_size':16,
    'eval_batch_size': 16,
    'max_seq_length':142,
    'learning_rate': 1e-5,
    'evaluate_during_training': True,
    'evaluate_during_training_steps': 20000,
    'overwrite_output_dir': True,
    'eval_all_checkpoints': True,
    'weight_decay': 0,
    'manual_seed' : 11,
    'optimizer' : 'AdamW',
    'adam_epsilon': 1e-8,
    'dropout': 0.3,
    'logging_steps': 50,
    'save_steps': 2000,
    'no_cache' : True,
    'wandb_project' :"RoBERTa_Fifty"})

# Train the model with training data and evaluating using eval data to prevent overfitting
model.train_model(train_df, eval_df=eval_df)
```

When the execution starts, the user will be asked to authenticate the wandb account. By clicking on the mentioned and logging in, user can get the API key. After entering the key the model training will begin.

```
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter: ··········
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.12.0
Syncing run royal-cherry-6 to Weights & Biases (Documentation).
```



Testing the classifier using test data

```
# Verifying the classifier using test data
predictions, raw_outputs = model.predict(test_df.cleaned_reviews.tolist())
```

Printing Confusion Matrix and Classification report

```
# Printing confusion matrix
matrix = confusion_matrix(y_test,predictions, labels=[0,1])
print('Confusion matrix : \n',matrix)
```

```
Confusion matrix :
 [[24412  8200]
 [ 8877 13362]]
```

```
#Printing Classification Report
matrix = classification_report(y_test,predictions,labels=[0,1])
print('Classification report : \n',matrix)
```

```
Classification report :
              precision    recall  f1-score   support

           0       0.73      0.75      0.74     32612
           1       0.62      0.60      0.61     22239

    accuracy                           0.69     54851
   macro avg       0.68      0.67      0.68     54851
weighted avg       0.69      0.69      0.69     54851
```

Similarly, BERT, DistilBERT, and ALBERT with 10% and 50% data are trained by making necessary changes in ClassificationModel and providing the values of hyper-parameters.

References

Mukherjee, A., Venkataraman, V., Liu, B., Glance, N. et al. (2013). Fake review detection: Classification and analysis of real and pseudo reviews, UIC-CS-03-2013. Technical Report.