National College of Ireland

Bitcoin Tweets Sentiment Analysis using Bidirectional Encoding Representational Transformers (BERT)

MSc Research Project
Data Analytics

Samir Huseynov
Student ID: x21245070

School of Computing
National College of Ireland

Supervisor: Bharat Agarwal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Samir Huseynov
Student ID:	x21245070
Programme:	Data Analytics
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Bharat Agarwal
Submission Due Date:	31/01/2024
Project Title:	Bitcoin Tweets Sentiment Analysis using Bidirectional Encoding Representational Transformers (BERT)
Word Count:	918
Page Count:	6

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:	Samir Huseynov
Date:	30 January 2024

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):	

Bitcoin Tweets Sentiment Analysis using Bidirectional Encoding Representational Transformers (BERT)

Configuration Manual

Introduction

A configuration manual contains step-by-step instructions for configuring a device or system. The goal of the manual is to provide a comprehensive guide on how to carry out the research study. It also details the machine setup needed to compile and execute the models. The processes involve installing the necessary programs and packages in addition to the minimal configuration that is advised for a project to succeed.

Project Files

In this project, the Jupiter lab is used for model training and evaluation, while the Jupyter notebook is utilized for data preparation, analysis, and exploration.

- Fetch the .CSV file of Bitcoin Tweets from Kaggle and perform the necessary data analysis Jupyter notebook.
- Fetch the .CSV file of Bitcoin Tweets from Kaggle. After data cleaning and preprocessing of raw data, apply the machine learning and deep learning models and evaluate the results in Jupyter Lab.

System Specification

A documented account of a system's technical specifications and requirements is called a system specification. This document describes the hardware requirements for a workstation that can handle intensive workloads in Deep Learning (Ain et al., 2017) and Machine Learning (Song et al., 2017), especially when big language models like BERT(Devlin et al., 2018) are involved. Efficient training and inference of complicated models is ensured by the system's robust setup that is tailored to meet the demanding computational requirements of these jobs. Figure 1 shows the technical specifications of the system.

Device Specifications	
Device Name	DESKTOP-2V7AG3I
Processor	Intel(R) Core (TM) i7-8700 CPU @ 3.20GHz 3.19 GHz
Installed RAM	16GB
Device ID	55EDB7-4AC2-4F0A-A7A6-3E6C41DBA8
Product ID	00330-80000-00000-AA414
System ID	64-bit Operating System, x64-based processor
Pen and Touch	No Pen or touch input is available for the display

Figure 1: System specifications

This workstation's powerful combination includes an 8th Gen Intel Core i7 processor for parallel processing, 16GB of DDR4 RAM to support large language models, a lightning-fast 1TB SSD to reduce latency in data access, and a dedicated NVIDIA GTX 100 Ti GPU to speed up computationally demanding tasks like matrix multiplication. Figure 2 shows the Operating system details used to work with machine learning and deep learning algorithms.

Window Specifications	
Edition	Window 10 Pro
Version	22Hz
Installed On	9/6/2023
OS build	19045.3693
Experience	Window Features Experience Pack 1000.19053.1000.0

Figure 2: Operating System Details

Tools and technologies

- Microsoft excel: Used for opening and exploring .CSV files.
- Anaconda: Used to create virtual environments for projects and installing packages.
- Jupyter Notebook: Used for Exploratory data analysis.
- Jupyter Lab: Used for data preprocessing, data cleaning, model traning and evaluation.

Environment Setup

Python needs to be installed initially, depending on the operating system; it is

recommended to install the most recent version[1]. The most recent version of the file was downloaded and installed, which was Python 3.9.1 for Windows 10. After installing Python, a development environment is required in order to write, execute, and see the results of code. Probably the most popular and easiest to use platform is Jupyter Notebook. It comes pre-installed with the Anaconda Python distribution[2], for which the appropriate installation can be downloaded based on the system. Figure 3 shows the Anaconda interface together with pre-installed programs such as the Jupyter notebook.

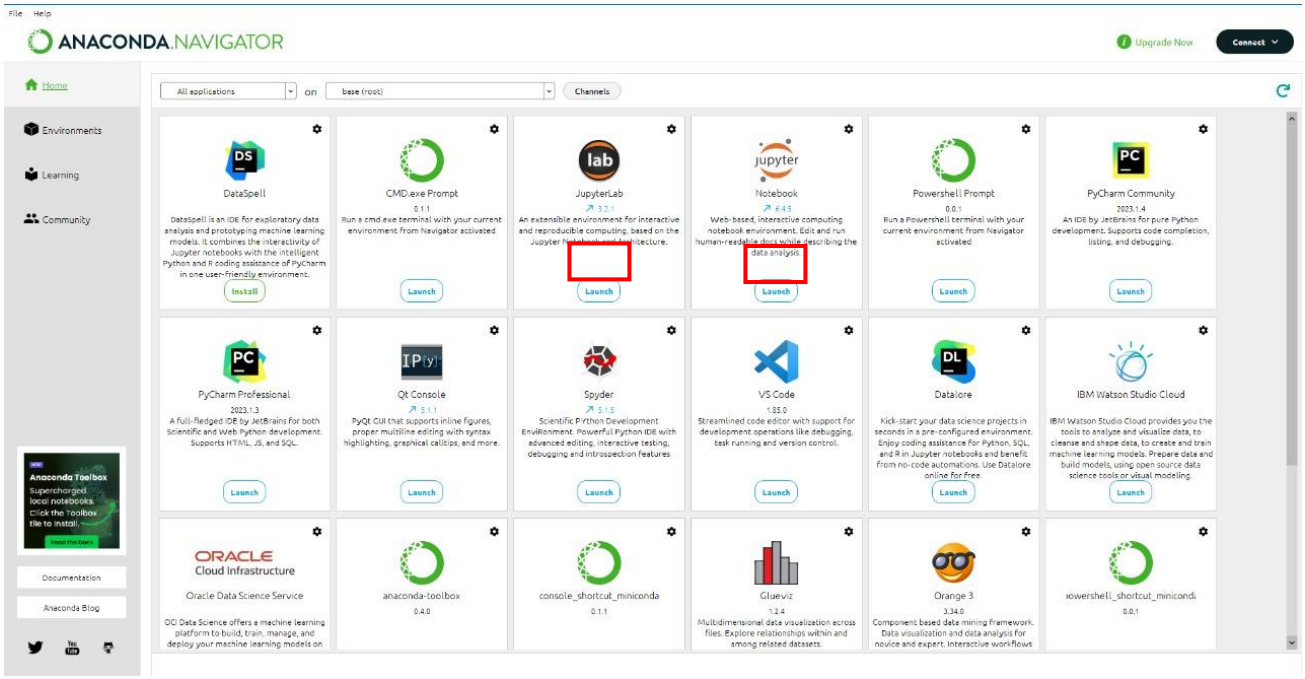

Figure 3: Anaconda GUI

Once all of these steps are finished, you may open the Jupyter notebook or JupyterLab by clicking on “launch” button.

Click the “new” button on top of the file to create a new notebook. Using the file reference

[1] https://www.python.org/downloads/

[2] https://www.anaconda.com/download

given in the code section, you can also choose to open a notebook file. If you need to install a package or library in the notebook, use the command "pip install package-name."

Importing libraries

There are different libraries or packages available in python language to perform a specific task, like image classification, text sentiment analysis, speech recognition etc. Every specific task in python language requires some libraries to use different classes and function which helps us to perform complex task and provide code reusability. Figure 4 shows different libraries need to import in Jupyter notebook for data preprocessing, model traning and evaluation

```
In [ ]: import os
        import nltk
        import torch
        import string
        import string
        import random
        import numpy as np
        import numpy as np
        import pandas as pd
        import pandas as pd
        import seaborn as sns
        from sklearn.svm import SVC
        from tabulate import tabulate
        from tabulate import tabulate
        from joblib import dump, load
        import matplotlib.pyplot as plt
        import matplotlib.pyplot as plt
        from nltk.corpus import stopwords
        from sklearn.metrics import f1_score
        from transformers import BertTokenizer
        from tqdm.notebook import trange, tqdm
        from nltk.stem import WordNetLemmatizer
        from torch.utils.data import TensorDataset
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.preprocessing import LabelEncoder
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.model_selection import train_test_split
        from transformers import BertForSequenceClassification
        from transformers import AdamW, get_linear_schedule_with_warmup
        from torch.utils.data import DataLoader, TensorDataset, SequentialSampler
        from torch.utils.data import DataLoader, RandomSampler, SequentialSampler
        from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
        from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score, f1_score
        from sklearn.metrics import confusion_matrix,classification_report, accuracy_score,precision_score, recall_score, f1_score
```

Figure 4: Libraries

Importing Dataset

To load the dataset in Jupyter notebook, use “read_csv” method in “pandas” library. This method will convert the .csv file into a data frame in Jupyter notebook.

Data Preprocessing

Figure 6 Shows the data preprocessing steps in Jupyter notebook:

- **Handle missing values:** To fill up data gaps without skewing conclusions, use techniques like encoding or imputation.
- **Removal of stopwords:** Removing common, insignificant terms to speed up text processing and concentrate on important information.
- **Lemmatization:** simplifying words to their most basic meaning and standardizing word nuances to facilitate comparison and analysis.
- **Stemming:** Crudely chopping words to their base forms for approximate meaning groupings.
- **Removal of punctuations:** To simplify text, concentrate on word content, and get data ready for processing, NLP employs punctuation removal, which eliminates punctuation such as commas, periods, and question marks.

```
dftaxes = df
dftaxes['Text_Data'] = dftaxes['Tweet'].apply(
    lambda x: " ".join(x.lower() for x in x.split()))  # Lower case conversion
dftaxes['Text_Data'] = dftaxes['Text_Data'].str.replace('[^\w\s]', '')  # getting rid of special characters
dftaxes['Text_Data'] = dftaxes['Text_Data'].str.replace('\d+',
                                                        '')  # removing numeric values from between the words
dftaxes['Text_Data'] = dftaxes['Text_Data'].apply(
    lambda x: x.translate(string.digits))  # removing numerical numbers
stop = stopwords.words('english')
dftaxes['Text_Data'] = dftaxes['Text_Data'].apply(
    lambda x: " ".join(x for x in x.split() if x not in stop))  # removing stop words
stemmer = WordNetLemmatizer()
dftaxes['Text_Data'] = [stemmer.lemmatize(word) for word in
                        dftaxes['Text_Data']]  # converting words to their dictionary form
dftaxes['Text_Data'] = dftaxes['Text_Data'].str.replace('shall', '')

# Feature extraction using TF-IDF
tfidf_vectorizer = TfidfVectorizer()
tfidf_features = tfidf_vectorizer.fit_transform(dftaxes)
print("\nTF-IDF Features:")
print(tfidf_features.toarray())
```

Figure 5: Data Preprocessing

Modeling

After applying preprocessing techniques, Extract the features from Tweets using “Count Vectorizer” or “Term Frequency method” (Deepa et al., 2019). Then, split the data into traning and testing sets. The traning set will be used to train the model while testing test is used to evaluate the performance of the model. Figure 7 shows the feature extraction and Figure 8 shows the model traning.

```
cv=CountVectorizer(ngram_range=(1,2))
X_cv=cv.fit_transform(df['Text_Data'])
X=X_cv
y=df['label']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=123)
```

Figure 6: Feature extraction

```
from transformers import BertForSequenceClassification

model = BertForSequenceClassification.from_pretrained('bert-base-uncased',
     num_labels=len(label_dict),
     output_attentions=False,
     output_hidden_states=False)
```

Figure 7: Modeling

Evaluation

After traning the model, evaluate your model using following matrices (Bekkar et al., 2013). Figure 9 shows the code of model evaluation.

- Accuracy
- Precision
- Recall
- F1-score

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix,classification_report, accuracy_score,precision_score, recall_score, f1_score
# function for evaluation metrics precision, recall, f1 etc
def modelEvaluation(predictions, y_test_set, model_name, classes):
    # Print model evaluation to predicted result
    print("==========",model_name,"==========")
    print ("\nAccuracy on validation set: {:.4f}".format(accuracy_score(y_test_set, predictions)))
    print ("\nClassification report : \n", classification_report(y_test_set, predictions, target_names = classes))
    print ("\nConfusion Matrix : \n", confusion_matrix(y_test_set,predictions))
    sns.heatmap(confusion_matrix(y_test_set, predictions),annot=True,yticklabels=classes, xticklabels=classes, fmt='g',cmap='jet

    plt.tight_layout()
    plt.show()
    results = [accuracy_score(y_test_set, predictions), precision_score(y_test_set, predictions,average='macro'),
               recall_score(y_test_set, predictions,average='macro'),f1_score(y_test_set, predictions,average='macro')]#store resu
    return results
```

Figure 8: Model Evaluation Measures

References

Ain, Q. T., Ali, M., Riaz, A., Noureen, A., Kamran, M., Hayat, B., & Rehman, A. (2017). Sentiment Analysis Using Deep Learning Techniques: A Review. *IJACSA) International Journal of Advanced Computer Science and Applications*, *8*(6). www.ijacsa.thesai.org

Bekkar, M., Kheliouane Djemaa, D., & Akrouf Alitouche, D. (2013). *Evaluation Measures for Models Assessment over Imbalanced Data Sets*. *3*(10). www.iiste.org

Deepa, D., Raaji, & Tamilarasi, A. (2019). Sentiment Analysis using Feature Extraction and Dictionary-Based Approaches. *Proceedings of the 3rd International Conference on I-SMAC IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2019*, 786–790. https://doi.org/10.1109/I-SMAC47947.2019.9032456

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*, 4171–4186. https://arxiv.org/abs/1810.04805v2

Song, C., Ristenpart, T., & Shmatikov, V. (2017). Machine learning models that remember too much. *Proceedings of the ACM Conference on Computer and Communications Security*, 587–601. https://doi.org/10.1145/3133956.3134077