National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Himani Chauhan
StudentID:x22118021

School of Computing
National College of Ireland

Supervisor: Mr Teerath Kumar Menghwar

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Himani Chauhan
Student ID:	x22118021
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Mr Teerath Kumar Menghawar
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	8

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:	13/12 2023

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

Himani Chauhan
x22118021

1 Introduction

This configuration manual explains the entire project's procedure in detail, from environment setup to implementation and assessment. The purpose of this project was to perform advanced sentiment analysis and topic modeling on e-commerce reviews from Flipkart and Amazon. The programming language that is used, the system settings, and any necessary libraries are all covered in this setup manual.

The results of this study, the many experiments carried out, and the evaluation standards applied to each experiment are the main topics of discussion.

2 Environment Setup

2.1 System Specification

Jupyter notebook was used i to carry out the implementation of this study. It is used to create and share documents with live code, equations, graphs, and narrative prose using the open-source, interactive online application Jupyter Notebook. Because of its versatility and usability, it is frequently used in data science, machine learning, and scientific research.

2.2 System Specification

Python language was used in this project, with all the packages and libraries listed below.

- NumPy
- Matplotlib
- Pandas
- NLTK
- pytorch
- tensorflow
- Seaborn
- Scikit-Learn
- Transformers
- Bert and Roberta
- LabelEncoder
- AdamWOptimizer

2.3 Data Source

The datasets used in this project are downloaded from Kaggle. Two datasets related to e-commerce reviews of widely known platforms called Flipkart and Amazon are downloaded.

3 Implementation

In this section, all the steps are showcased right from start to end of the implementation proposed in this project. We are showcasing all the Libraries loading, data preparation and implementing models.

3.1 Importing Libraries

In figure 1. all the necessary libraries are imported for this project.

```
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns

from wordcloud import WordCloud
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
import datetime
```

Figure 1 Importing Libraries

3.2 Data Loading and Pre-processing

In figure 2. and 3 data loading process is done for both the topics: sentiment analysis and Topic Modeling. Data Pre-processing was performed to remove non-English words, URLs, stop words, removing duplicates, and normalizing data and null values etc. Below snippets show some of the steps carried out to input data.

```
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
import datetime


# Load the data
amazon_data = pd.read_csv('7817_1.csv')
flipkart_data = pd.read_csv('flipkart_product.csv', encoding='latin1')
```

Figure 2 Loading the data

```
# Remove duplicates
amazon_data.drop_duplicates(subset='reviews.text', inplace=True)
flipkart_data.drop_duplicates(subset='Review', inplace=True)

# Filter out non-English reviews (this is a simple approach and might not catch all non-English reviews)
amazon_data = amazon_data[amazon_data['reviews.text'].apply(lambda x: bool(re.search('[a-zA-Z]', str(x))))]
flipkart_data = flipkart_data[flipkart_data['Review'].apply(lambda x: bool(re.search('[a-zA-Z]', str(x))))]

# Text Normalization
def preprocess_text(text):
    # If text is not a string type, return an empty string or convert to string
    if not isinstance(text, str):
        return ""

    # Convert to lowercase
    text = text.lower()

    # Remove URLs
    text = re.sub(r'http\S+', '', text)

    # Remove punctuations
    text = re.sub(r'[^\w\s]', '', text)
    return text

amazon_data['processed_review'] = amazon_data['reviews.text'].apply(preprocess_text)
flipkart_data['processed_review'] = flipkart_data['Review'].apply(preprocess_text)
```

Figure 3 Data Preprocessing

3.3 Exploratory Data Analysis

Preliminary exploratory data analysis is performed on both the datasets to get more insights about dataset.

```
# Check for missing values
print(amazon_data.isnull().sum())

# Distribution of ratings
plt.figure(figsize=(8, 6))
sns.countplot(x='reviews.rating', data=amazon_data, palette='viridis')
plt.title('Distribution of Ratings')
plt.show()

# Distribution of review lengths
amazon_data['review_length'] = amazon_data['processed_review'].apply(len)
plt.figure(figsize=(10, 6))
sns.histplot(amazon_data['review_length'], bins=30, kde=True, color='skyblue')
plt.title('Distribution of Review Lengths')
plt.xlabel('Review Length')
plt.ylabel('Frequency')
plt.show()

# Word cloud for reviews
from wordcloud import WordCloud

# Concatenate all reviews into a single string
all_reviews = ' '.join(amazon_data['processed_review'].astype(str))

# Generate word cloud
wordcloud = WordCloud(width=800, height=400, random_state=42, max_font_size=110).generate(all_reviews)

# Display the word cloud
plt.figure(figsize=(10, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Amazon Reviews')
plt.show()
```

Figure 4 Exploratory Data Analysis

3.4 Sentiment Analysis using VADER

We are creating a baseline model for sentiment analysis on both Flipkart and Amazon dataset which will lay foundation for our advanced models. Below snippet shows the code and results of the methods applied.

```
# Apply VADER to each review
flipkart_data['vader_sentiment'] = flipkart_data['processed_review'].apply(lambda x: analyzer.polarity_scores(x)['compound'])
true_labels = flipkart_data['sentiment']
vader_sentiment = flipkart_data['vader_sentiment']
```

```
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder

# Convert categorical labels to numeric using LabelEncoder
label_encoder = LabelEncoder()
true_labels_encoded = label_encoder.fit_transform(true_labels)

# Apply threshold to VADER sentiment scores (replace 0.5 with your threshold)
vader_sentiment_thresholded = (vader_sentiment > 0.5).astype(int)

# Print classification report
print("VADER Sentiment Analysis Report:")
print(classification_report(true_labels_encoded, vader_sentiment_thresholded, target_names=label_encoder.classes_))
```

```
VADER Sentiment Analysis Report:
              precision    recall  f1-score   support

    negative       0.28      1.00      0.43       237
     neutral       0.02      0.08      0.03       111
    positive       0.00      0.00      0.00       911

    accuracy                           0.19      1259
   macro avg       0.10      0.36      0.16      1259
weighted avg       0.05      0.19      0.08      1259
```

Figure 5 Sentiment Analysis using VADER

```
Accuracy: 0.00
Classification Report:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00       0.0
     Neutral       0.00      0.00      0.00       0.0
    Positive       0.00      0.00      0.00       0.0
         nan       0.00      0.00      0.00     150.0
    negative       0.00      0.00      0.00      72.0
     neutral       0.00      0.00      0.00      69.0
    positive       0.00      0.00      0.00     757.0

    accuracy                           0.00    1048.0
   macro avg       0.00      0.00      0.00    1048.0
weighted avg       0.00      0.00      0.00    1048.0

Confusion Matrix:
[[  0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0]
 [ 22   1 127   0   0   0   0]
 [ 31   6  35   0   0   0   0]
 [ 11   3  55   0   0   0   0]
 [ 36  32 689   0   0   0   0]]
```

3.5 Sentiment Analysis using BERT

The next model built is sentiment analysis using BERT embeddings on both Flipkart and Amazon Datasets. A methodical approach to fine-tune the BERT model for multi-class

sentiment classification is to use Sentiment Analysis on the Flipkart Dataset using BERT. The BERT tokenizer and sequence classification model are initially implemented using the Hugging Face Transformers library.

```
# Training loop
optimizer = AdamW(model.parameters(), lr=2e-5)

# Fine-tune the model
for epoch in range(3):  # Adjust the number of epochs based on your project
    model.train()
    for batch in train_dataloader:
        optimizer.zero_grad()
        input_ids, labels = batch
        outputs = model(input_ids, labels=labels)

        # Convert labels to the correct data type
        labels = labels.type(torch.LongTensor)

        loss = outputs.loss
        loss.backward()
        optimizer.step()

# Evaluation
model.eval()
predictions = []
true_labels = []
with torch.no_grad():
    for batch in test_dataloader:
        input_ids, labels = batch
        outputs = model(input_ids)
        logits = outputs.logits
        predictions.extend(torch.argmax(logits, dim=1).tolist())
        true_labels.extend(labels.tolist())

# Calculate metrics
print("BERT Sentiment Analysis Report:")
print(classification_report(true_labels, predictions))
```

Figure 6 Sentiment Analysis using BERT

BERT Sentiment Analysis Report:

	precision	recall	f1-score	support
0	0.58	0.88	0.70	16
1	1.00	0.06	0.12	16
2	0.89	0.96	0.92	95
accuracy			0.83	127
macro avg	0.83	0.63	0.58	127
weighted avg	0.87	0.83	0.79	127

Figure 7 Flipkart Sentiment Analysis

BERT Sentiment Analysis Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	28
1	0.00	0.00	0.00	11
2	0.67	0.95	0.78	63
3	0.00	0.00	0.00	4
accuracy			0.57	106
macro avg	0.17	0.24	0.20	106
weighted avg	0.40	0.57	0.47	106

Figure 8 Amazon Sentiment Analysis

3.6 Sentiment Analysis using RoBERTa

Utilizing state-of-the-art language models, a systematic and detailed method is used to execute sentiment analysis using RoBERTa on the Flipkart dataset. First, the code loads the pre-trained RoBERTa tokenizer and model using the Hugging Face Transformers library. Tokenization and encoding are performed on Flipkart dataset, the reviews have been preprocessed and stored in column 'processedreview', which contains the cleaned and tokenized reviews.

```
# Fine-tune RoBERTa model
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
model = RobertaForSequenceClassification.from_pretrained('roberta-base', num_labels=label_encoder.classes_.shape[0])

# Tokenize and encode the training data
encoded_reviews = [tokenizer.encode(review, add_special_tokens=True, truncation=True, padding=True) for review in train_data['p
max_len = max(len(review) for review in encoded_reviews)
padded_reviews = [review + [0] * (max_len - len(review)) for review in encoded_reviews]
input_ids = torch.tensor(padded_reviews)

# Encode the labels
labels = torch.tensor(train_data['label'].values)

# Create TensorDataset
train_dataset = TensorDataset(input_ids, labels)

# Initialize DataLoader
train_dataloader = DataLoader(train_dataset, batch_size=4, shuffle=True)

# Training loop
optimizer = AdamW(model.parameters(), lr=2e-5)

# Fine-tune the model
for epoch in range(3):  # Adjust the number of epochs based on your project
    model.train()
    for batch in train_dataloader:
        optimizer.zero_grad()
        input_ids, labels = batch
        outputs = model(input_ids, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
```

Figure 9 Sentiment Analysis usin RoBERTa

RoBERTa Sentiment Analysis Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	16
1	0.00	0.00	0.00	16
2	0.75	1.00	0.86	95
accuracy			0.75	127
macro avg	0.25	0.33	0.29	127
weighted avg	0.56	0.75	0.64	127

Figure 10 Flipkart Sentiment Analysis Report

RoBERTa Sentiment Analysis Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	9
1	0.00	0.00	0.00	2
2	0.80	0.95	0.87	83
3	0.17	0.09	0.12	11
accuracy			0.76	105
macro avg	0.24	0.26	0.25	105
weighted avg	0.65	0.76	0.70	105

Figure 11 Amazon Sentiment Analysis Report

3.7 Topic Modeling using BERT

In this section we are using BERT embeddings and K-Means clustering to do topic modeling on the Flipkart dataset. Tokenizing the preprocessed reviews using the BERT tokenizer at the start of the procedure gives encoded sequences.

```
# Tokenize the processed reviews using BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
encoded_reviews = [tokenizer.encode(review, add_special_tokens=True, truncation=True, padding=True) for review in amazon_data['pr

# Find the maximum length among the encoded reviews
max_len = max(len(review) for review in encoded_reviews)

# Pad all the sequences to the maximum length
padded_reviews = [review + [0] * (max_len - len(review)) for review in encoded_reviews]

# Convert the lists of lists to a PyTorch tensor
input_ids = torch.tensor(padded_reviews)

# Load the pre-trained BERT model
model = BertModel.from_pretrained('bert-base-uncased')

# Get the embeddings for each review
with torch.no_grad():
    model.eval()
    embeddings = model(input_ids)[0][:, 0, :].numpy()  # Extract embeddings from the [CLS] token

# Perform K-Means clustering for topic modeling
num_topics = 5  # Adjust based on your desired number of topics
kmeans = KMeans(n_clusters=num_topics, random_state=42)
topics = kmeans.fit_predict(embeddings)

# Display the reviews for each topic
amazon_data['topic'] = topics
for topic_idx in range(num_topics):
    topic_reviews = amazon_data[amazon_data['topic'] == topic_idx]['processed_review']
    print(f"Topic #{topic_idx} Reviews:")
    print('\n'.join(topic_reviews))
    print('\n' + '='*50 + '\n')
```

3.8 Topic Modeling using RoBERTa

```
# Load the pre-trained RoBERTa model
model = RobertaModel.from_pretrained('roberta-base')

# Get the embeddings for each review
with torch.no_grad():
    model.eval()
    embeddings = model(input_ids)[0]

# Flatten the embeddings
flattened_embeddings = embeddings.view(embeddings.size(0), -1)

# Assuming flattened_embeddings contains your BERT or RoBERTa embeddings
# Ensure all values are non-negative by adding the absolute minimum value
min_val = flattened_embeddings.min()
flattened_embeddings -= min_val

# Perform Latent Dirichlet Allocation (LDA) for topic modeling
lda = LatentDirichletAllocation(n_components=5, random_state=42)
topics = lda.fit_transform(flattened_embeddings)



# Display the top words for each topic
feature_names = [f"feature_{i}" for i in range(len(flattened_embeddings[0]))]
for topic_idx, topic in enumerate(lda.components_):
    top_features_idx = topic.argsort()[:-10 - 1:-1]
    top_words = [tokenizer.decode([i]) for i in top_features_idx]
    print(f"Topic #{topic_idx}: {', '.join(top_words)}")
```

```
Topic #0:  160,  Watts,  real,  Disc,  supremacist,  consequence,  statistics,  Byrne,  Cond,  separatist
Topic #1:  DA, United, estead,  real,  arrangement,  ignorance,  statistics,  disorders, grad,  Oaks
Topic #2:  railways, United,  Lash,  supremacist, estead,  warning,  sounds, grad,  surgeries,  injury
Topic #3:  Watts,  consequence,  noting,  separatist,  ignorance,  Lash,  Sn,  injury,  sounds,  Supreme
Topic #4:  Oaks,  semester,  Patterson,  arrangement,  Lash,  wheel, UG,  disorders,  stole,  separatist
```