Configuration Manual

MSc Research Project
Msc in Artificial Intelligence

Madhuri Chowdappa Madhugiri
Student ID:X22200622

School of Computing
National College of Ireland

Supervisor: Kislay Raj

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Madhuri Chowdappa Madhugiri
Student ID:	X22200622
Programme:	MSc in Artificial intellegence **Year:** 2023-2024
Module:	MSc Research Project
Lecturer:	Kislay Raj
Submission Due Date:	12/08/2024
Project Title:	Aspect Based Sentiment Analysis using Pretrained Model…
Word Count:	980 **Page Count:** 11

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: Madhuri

Date: 12/08/2024

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):	

Configuration Manual

Madhuri Chowdappa Madhugiri
Student ID: x22200622

1 Introduction

This configuration manual provides the comprehensive guide to set up the environment, installing the required software, configuring necessary tools and managing datasets for the successful replication of the experimental setup for Aspect-Based Sentiment Analysis using a combination of BERT and Hierarchical Attention Network (HAN) with Knowledge Graphs. This manual is designed to assist to replicate the experiments described in the project. It does not cover the installation of standard software but focuses on the specific configurations and settings that are essential for this project.

2 System Configuration

- **Processor:** 12th Gen Intel(R) Core(TM) i5-1235U @ 1.30 GHz
- **RAM:** 16.0 GB
- **System Type:** 64-bit Operating System, x64-based Processor
- **Operating System:** Windows 11 Home Single Language

This setup provides sufficient computational power and memory to efficiently run the Python scripts and machine learning models used in this project. The use of a GPU is recommended for faster training times, particularly when working with deep learning models like BERT and HAN.

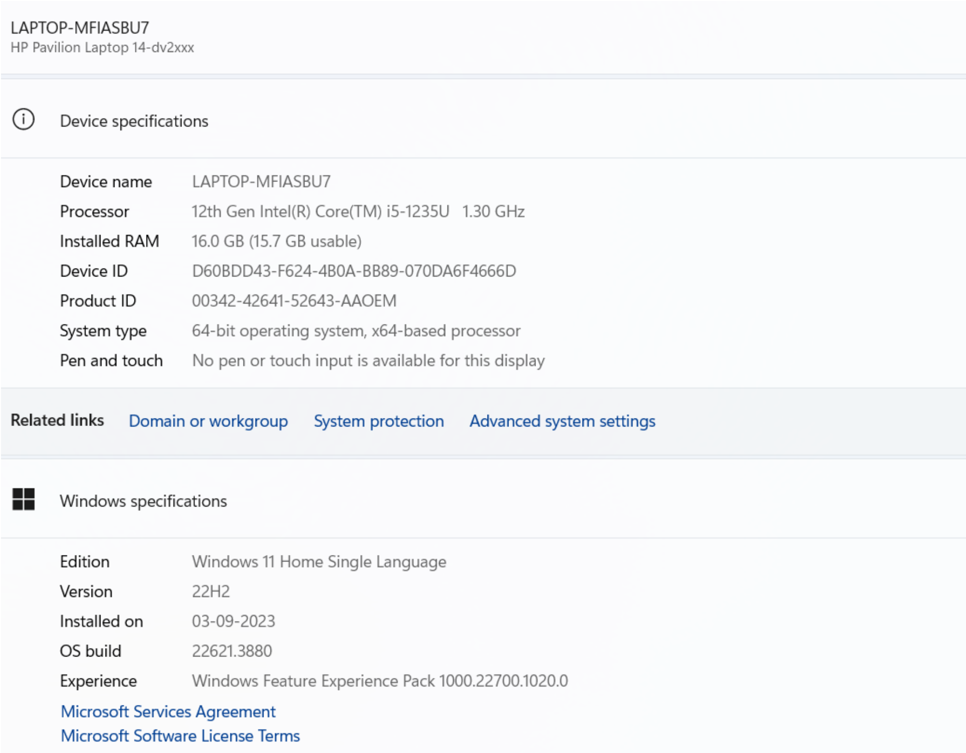

Fig 1: System Configuration

3 Software Requirements

To carry out the project ensure that the following software tools are installed on your system:

1. **Anaconda** - Used for managing Python environments and packages. Version 2.3.3 or later is recommended.
2. **Python** - Version 3.9.7, which comes with Anaconda distribution.
3. **Jupyter Notebook** - Used for interactive development and testing of code.
4. **Google Colab** - Optionally used for running scripts on a cloud-based environment with GPU acceleration.

4 Python Libraries

The project uses the several Python libraries for data processing, machine learning and visualization:

1. pandas
2. matplotlib
3. seaborn
4. nltk

5. spacy
6. gensim
7. scikit-learn
8. transformers
9. torch (PyTorch)
10. pyLDAvis
11. lime
12.

The required libraries can be installed using pip with the following command:

pip install pandas matplotlib seaborn nltk spacy gensim scikit-learn transformers torch scipy pyLDAvis lime

After installing the libraries, make sure to download the required Spacy model by running:

python -m spacy download en_core_web_sm

This will set up all the necessary Python libraries required to execute the provided code.

5 Filepaths Configuration

5.1 Local Machine

Dataset Path Configuration:

```
import pandas as pd

file_path = 'cleaned_data.csv'

df = pd.read_csv(file_path)

df
```

Fig 5.1 Dataset Path Configuration in Local Machine

Adjust file paths to point to your local dataset.

3.2 Google Colab

Mount Google Drive:

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

    Mounted at /content/drive
```

```
import pandas as pd

# Example path
file_path = '/content/drive/My Drive/final_dataset.csv'

# Read CSV file
data = pd.read_csv(file_path)

# Show the DataFrame
print(data)
df = pd.DataFrame(data)

        Rating                                                Review
```

Fig 5.2 Dataset Configuration in Google Colab

6 Dataset

The dataset used in this project is a combination of reviews from three sources:

1. Yelp
2. TripAdvisor
3. Amazon Product Reviews

These individual datasets are combined into the single dataset and saved as final_dataset.csv. The combined dataset contains the reviews from various domains which provides the rich and diverse set of data for sentiment analysis.

Steps:

Loading the Dataset:

The dataset used in this project is the combination of the reviews from three sources.

Preprocessing:

After loading the dataset preprocessing is carried out to clean and standardize the text. This processed data is saved as cleaned_and_preprocessed_data.csv.

Sentiment Analysis on Aspects:

Sentiment analysis is performed on the extracted aspects and the final dataset is saved as dataset_with_aspects.csv.

Note: The dataset files will be provided. Ensure that file paths are adjusted in the code to correctly point to these files in local environment/colab.

7 Data Preprocessing

Before running the models, the dataset undergoes various preprocessing steps:

1. **Loading the Dataset:** Ensure that the final dataset is loaded from final_dataset.csv in your notebook or IDE.
2. **Text Preprocessing:** This involves tokenization, removing stopwords, and lemmatization to clean the text data.
3. **Handling Class Imbalance:** Resample the minority class to match the majority class size.

	Rating	Review	review_cleaned	Sentiment	Review_length
0	5	This is a great, reliable phone. I also purcha...	['great', 'reliable', 'phone', 'also', 'purcha...	1	768
1	4	The phone has been great for every purpose it ...	['phone', 'great', 'every', 'purpose', 'offer'...	1	606
2	4	Hello, I have this phone and used it until I d...	['hello', 'phone', 'used', 'decided', 'buy', '...	1	388
3	5	Cool. Cheap. Color: 3 words that describe the ...	['cool', 'cheap', 'color', 'word', 'describe',...	1	911
4	4	The 3599i is overall a nice phone, except that...	['overall', 'nice', 'phone', 'except', 'nokia'...	1	356
...	...	...	...	...	...
58139	3	no heat room marriott currently undergoing con...	['heat', 'room', 'marriott', 'currently', 'und...	0	426
58140	1	place pretty bad, rooms damp dirty, water line...	['place', 'pretty', 'bad', 'room', 'damp', 'di...	0	106
58141	2	I'd rate this below average mexican food. My ...	['rate', 'average', 'mexican', 'food', 'fajita...	0	244
58142	1	I don't think this is a Nokia lumia 900, becau...	['think', 'nokia', 'lumia', 'gorilla', 'glass'...	0	313
58143	2	I had to write an order after coming here a co...	['write', 'order', 'coming', 'couple', 'time',...	0	658

58144 rows × 5 columns

Fig 7.1 Cleaned and processed Data

4. Aspect Sentiment Analysis Using BERT: After preprocessing the BERT model is used to perform the sentiment analysis on the extracted aspects. The output is then saved as “dataset_with_aspects.csv”.

	Rating	Review	review_cleaned	Sentiment	Review_length	Aspect_Sentiments
0	5	This is a great, reliable phone. I also purcha...	[great, reliable, phone, also, purchased, phon...	1	768	{'great': 'POSITIVE', 'reliable': 'POSITIVE', ...
1	4	The phone has been great for every purpose it ...	[phone, great, every, purpose, offer, except, ...	1	606	{'phone': 'NEGATIVE', 'great': 'POSITIVE', 'ev...
2	4	Hello, I have this phone and used it until I d...	[hello, phone, used, decided, buy, flip, phone...	1	388	{'hello': 'POSITIVE', 'phone': 'NEGATIVE', 'us...
3	5	Cool. Cheap. Color: 3 words that describe the ...	[cool, cheap, color, word, describe, nokia, pe...	1	911	{'cool': 'POSITIVE', 'cheap': 'NEGATIVE', 'col...
4	4	The 3599i is overall a nice phone, except that...	[overall, nice, phone, except, nokia, made, un...	1	356	{'overall': 'POSITIVE', 'nice': 'POSITIVE', 'p...
...	...	...	...	...	...	...
58139	3	no heat room marriott currently undergoing con...	[heat, room, marriott, currently, undergoing, ...	0	426	{'heat': 'POSITIVE', 'room': 'POSITIVE', 'marr...
58140	1	place pretty bad, rooms damp dirty, water line...	[place, pretty, bad, room, damp, dirty, water,...	0	106	{'place': 'POSITIVE', 'pretty': 'POSITIVE', 'b...
58141	2	I'd rate this below average mexican food. My ...	[rate, average, mexican, food, fajita, tasted,...	0	244	{'rate': 'POSITIVE', 'average': 'POSITIVE', 'm...
58142	1	I don't think this is a Nokia lumia 900, becau...	[think, nokia, lumia, gorilla, glass, bought, ...	0	313	{'think': 'POSITIVE', 'nokia': 'POSITIVE', 'lu...
58143	2	I had to write an order after coming here a co...	[write, order, coming, couple, time, place, te...	0	658	{'write': 'POSITIVE', 'order': 'POSITIVE', 'co...

58144 rows × 6 columns

Fig 7.2 Dataset after ading the Aspect_sentiments column

8 Model Training and Testing with BERT

Once the dataset with aspect sentiments is ready the BERT model is used for model training and evaluation:

1. **Model Preparation:**The BERT model is fine-tuned on the processed dataset to perform aspect-based sentiment classification.

```
# Load the pre-trained BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

def tokenize_data(data):
    return tokenizer(data.tolist(), padding=True, truncation=True, max_length=512, return_tensors='pt')

# Tokenize the data
X_train, X_test, y_train, y_test = train_test_split(aspect_df['input'], aspect_df['label'], test_size=0.2, random_state=42)
train_encodings = tokenize_data(X_train)
test_encodings = tokenize_data(X_test)

# Create PyTorch datasets
class SentimentDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

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

train_dataset = SentimentDataset(train_encodings, y_train.tolist())
test_dataset = SentimentDataset(test_encodings, y_test.tolist())
```

Fig 8.1 Code for model development

2. **Training:**The model is trained using a standard training loop, adjusting parameters to optimize performance.

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
)

trainer.train()
```

```
model.safetensors: 100% 440M/440M [00:02<00:00, 165MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized: ['classifier
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
<ipython-input-11-969b73e6540b>:19: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.
  item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
[1750/1750 46:52, Epoch 10/10]
```

Epoch	Training Loss	Validation Loss
1	0.516300	0.519913
2	0.154400	0.257172
3	0.188400	0.201714
4	0.047400	0.261489
5	0.096300	0.283864
6	0.000600	0.305707
7	0.001100	0.271117

Fig 8.2 Model Training

3. **Evaluation:**Evaluate the model using metrics such as accuracy, F1 score, precision, recall, and confusion matrices.

```
# Accuracy, Precision, Recall, F1-Score
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
f1 = f1_score(y_test, predictions)
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
```

```
<ipython-input-11-969b73e6540b>:19: UserWarning: To copy construct from a tensor, it is recomme
  item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
Evaluation Results: {'eval_loss': 0.2017143964767456, 'eval_runtime': 20.6318, 'eval_samples_pe
              precision    recall  f1-score   support

    Negative       0.92      0.79      0.85       148
    Positive       0.95      0.98      0.96       550

    accuracy                           0.94       698
   macro avg       0.93      0.89      0.91       698
weighted avg       0.94      0.94      0.94       698

[[117  31]
 [ 10 540]]
Accuracy: 0.9413
Precision: 0.9457
Recall: 0.9818
F1-Score: 0.9634
```

Fig 8.3 Code and Output for Model Evaluation

9 Hierarchical Attention Network (HAN) with Knowledge Graph and BERT

After evaluating the BERT model the project implements a Hierarchical Attention Network (HAN) combined with knowledge graph embeddings for enhanced aspect-based sentiment analysis:

1. **HAN Model Configuration:** The HAN model is designed to process word-level and sentence-level features of the text. The output is combined with knowledge graph embeddings to improve contextual understanding.

```
import torch.nn as nn
from transformers import BertModel

class HANWithKnowledgeGraph(nn.Module):
    def __init__(self, bert_model_name='bert-base-uncased', hidden_size=768, knowledge_size=300):
        super(HANWithKnowledgeGraph, self).__init__()
        self.bert = BertModel.from_pretrained(bert_model_name)
        self.word_attention = nn.MultiheadAttention(embed_dim=hidden_size, num_heads=8)
        self.sentence_attention = nn.MultiheadAttention(embed_dim=hidden_size, num_heads=8)
        self.fc = nn.Linear(hidden_size + knowledge_size, 1)  # Binary classification

    def forward(self, input_ids, attention_mask, knowledge_embedding):
        bert_output = self.bert(input_ids=input_ids, attention_mask=attention_mask)
        word_output, _ = self.word_attention(bert_output.last_hidden_state, bert_output.last_hidden_state, bert_output.last_hidden_state)
        sentence_output, _ = self.sentence_attention(word_output, word_output, word_output)
        pooled_output = torch.mean(sentence_output, 1)

        combined_output = torch.cat((pooled_output, knowledge_embedding), dim=1)
        logits = self.fc(combined_output)
        return logits
```

```
class AspectSentimentDatasetWithKnowledge(Dataset):
    def __init__(self, reviews, aspects, sentiments, tokenizer, max_length=512):
        self.reviews = reviews
        self.aspects = aspects
        self.sentiments = sentiments
        self.tokenizer = tokenizer
        self.max_length = max_length

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

    def __getitem__(self, idx):
        text = f"Review: {self.reviews[idx]} Aspects: {' '.join(self.aspects[idx])}"
        label = torch.tensor(self.sentiments[idx], dtype=torch.float)
        knowledge_embedding = get_knowledge_embedding_tensor(text)
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_length,
            truncation=True,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='pt'
        )
        return {
            'input_ids': encoding['input_ids'].squeeze(),
            'attention_mask': encoding['attention_mask'].squeeze(),
            'knowledge_embedding': knowledge_embedding.squeeze(),
            'labels': label.unsqueeze(0)
        }
```

Fig 9.1 Model Config for HAN with Knowledge graph

2. **Model Training:** HAN model trained using PyTorch using the knowledge graph embeddings for more accurate sentiment predictions.

```python
num_epochs = 3
han_model.train()

for epoch in range(num_epochs):
    total_loss = 0
    for batch in train_dataloader:
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        knowledge_embedding = batch['knowledge_embedding'].to(device)
        labels = batch['labels'].to(device).squeeze()  # Ensure labels have correct shape

        # Forward pass
        outputs = han_model(input_ids, attention_mask, knowledge_embedding).squeeze(-1)

        # Compute loss
        loss = criterion(outputs, labels)

        # Backward pass and optimization
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    avg_loss = total_loss / len(train_dataloader)
    print(f'Epoch {epoch + 1}/{num_epochs}, Loss: {avg_loss}')
```

3. **Evaluation:**Evaluate the HAN model using the same metrics as the BERT model and compare performance improvements.

```python
        all_preds.extend(preds.cpu().numpy())


cm = confusion_matrix(all_labels, all_preds)


if cm.size > 0:
    # Plot the confusion matrix
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Negative', 'Positive'])
    disp.plot(cmap=plt.cm.Blues)
    plt.title("Confusion Matrix")
    plt.show()
else:
    print("Confusion matrix is empty or invalid.")

Example usage
aluate_and_plot_confusion_matrix(han_model, val_dataloader, device)
```