National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Praneeth Gandham
Student ID:x23280891

School of Computing
National College of Ireland

Supervisor: Harshani Nagahamulla

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Praneeth Gandham
Student ID:	X23280891
Programme:	Msc Data Analytics c **Year:** 2024-25
Module:	Msc Research Practicum
Lecturer:	Harshani Nagahamulla
Submission Due Date:	11/08/25
Project Title:	Aspect Based Sentiment Analysis for Brand Monitoring: Handling Informal Language in App Review Data
Word Count:	1013 **Page Count: 12**

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:	Praneeth Gandham
Date:	10/08/25

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

Praneeth Gandham
Student ID: x23280891

1. Introduction

This configuration manual provides a comprehensive guide for replicating and configuring the Aspect-Based Sentiment Analysis (ABSA) system designed to classify sentiment in informal app review text. The implementation combines a transformer-based deep learning model (BERT) with a lexicon-based sentiment analysis method to enhance robustness in noisy, slang-filled, and emoji-rich text environments.

The manual explains dataset preparation, preprocessing, model training, evaluation, and hybrid ensemble construction, ensuring reproducibility in research and deployment contexts.

2. System Requirements

2.1. Hardware Requirements

- Processor: Multi-core CPU
- RAM: Minimum 8 GB (16 GB recommended for faster model training)
- GPU: CUDA-compatible GPU recommended
- Storage: Minimum 5 GB free space

2.2. Software Requirements

- Operating System: macOS, Linux, or Windows
- Python Version: 3.11
- Libraries:
    - pandas – Data manipulation
    - numpy – Numerical computation
    - matplotlib, seaborn – Data visualization
    - scikit-learn – Preprocessing, metrics, train-test splitting
    - nltk – Lexicon-based sentiment analysis (VADER)
    - transformers – HuggingFace BERT model and tokenizer
    - datasets – Dataset handling for Trainer API
    - torch – PyTorch backend for model training
    - accelerate – Optimized training support

Installation commands:

pip install pandas numpy matplotlib seaborn scikit-learn nltk torch transformers datasets accelerate emoji

3. Dataset Configuration

3.1. Dataset Location and Loading

The dataset train.csv must be stored in an accessible directory. The file should contain the following columns:

- **text**: User-generated review text
- **aspect**: Specific feature or topic referenced in the review
- **label**: Sentiment label (0 = Negative, 1 = Neutral, 2 = Positive)

Example loading command:

```
import pandas as pd
# Load the uploaded dataset
file_path = "/Users/gandhampraneeth/Desktop/Thesies/train.csv"
df = pd.read_csv(file_path)
```

4. Data Inspection

Upon loading, the dataset structure should be verified using:

```
# Display basic info
df_info = df.info()
df_head = df.head()
df_description = df.describe(include='all')
df_nulls = df.isnull().sum()

df_head, df_description, df_nulls
```

5. Data Preprocessing

5.1. Text Cleaning Function

A standardized cleaning pipeline is applied:

1. Convert text to lowercase.
2. Remove URLs, mentions (@), and hashtags (#).
3. Remove special characters and extra whitespace.
4. Normalize aspect terms to lowercase.

```
def clean_text(text):
    import re
    text = str(text).lower()
    text = re.sub(r"http\S+|www\S+", "", text)
    text = re.sub(r"\@\w+|\#", "", text)
    text = re.sub(r"[^a-zA-Z0-9\s]", "", text)
    text = re.sub(r"\s+", " ", text).strip()
    return text

df["clean_text"] = df["text"].apply(clean_text)
df["clean_aspect"] = df["aspect"].apply(lambda x: str(x).lower().strip())
```

5.2. Aspect Integration

For BERT compatibility, aspect terms are concatenated with the cleaned review using [SEP] as a separator:

```
# Combine aspect + text (as [aspect] [SEP] review)
df["combined_input"] = df["clean_aspect"] + " [SEP] " + df["clean_text"]
```

6. Exploratory Data Analysis (EDA)

6.1. Class Distribution

The dataset is moderately imbalanced (Negative > Neutral > Positive). This is visualized using:

```
# Label Mapping
label_map = {0: 'Negative', 1: 'Neutral', 2: 'Positive'}
df['label_text'] = df['label'].map(label_map)

# Visualizing Class Distribution
sns.countplot(x='label_text', data=df)
plt.title('Distribution of Sentiment Labels')
plt.show()
```

7. Train-Test Split

The dataset is split into:

- **Training set:** 80% of data
- **Test set:** 20% of data

Maintaining stratification preserves label distribution:

```
# Train/Test Split
train_texts, test_texts, train_labels, test_labels = train_test_split(
    df["combined_input"].tolist(), df["label"].tolist(), test_size=0.2, random_state=42
)
```

8. Transformer Model (BERT) Configuration

8.1. Tokenization

HuggingFace's BertTokenizer tokenizes, truncates, and pads sequences to a fixed length of 128 tokens:

```
# Tokenize
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=128)
test_encodings = tokenizer(test_texts, truncation=True, padding=True, max_length=128)
```

8.2. Dataset Preparation

The tokenized sequences are converted to datasets.Dataset objects for Trainer API compatibility.

```
# Convert to HuggingFace Dataset
train_dataset = Dataset.from_dict({
    'input_ids': train_encodings['input_ids'],
    'attention_mask': train_encodings['attention_mask'],
    'labels': train_labels
})

test_dataset = Dataset.from_dict({
    'input_ids': test_encodings['input_ids'],
    'attention_mask': test_encodings['attention_mask'],
    'labels': test_labels
})
```

8.3. Model Initialization

BERT is loaded with 3 output labels for sentiment classification:

```
# Load BERT model
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=3)
```

8.4. Training Arguments

Configured for 2 epochs, batch size 8, and logging every 10 steps:

```
# Define Trainer config
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=2,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    logging_dir="./logs",
    logging_steps=10,
    save_strategy="no"
)
```

8.5. Model Training

The HuggingFace Trainer API is used:

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

# Train model
trainer.train()
```

9. Lexicon-Based Model Configuration

9.1. Tools Used

```
# Lexicon-Based Sentiment Scoring

import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Download VADER only once
nltk.download("vader_lexicon")
vader = SentimentIntensityAnalyzer()

# Sample emoji and slang sentiment dictionaries
emoji_sentiment = {
    "😊": 1, "😃": 1, "😍": 1, "👍": 1,
    "😢": -1, "😠": -1, "😡": -1, "👎": -1,
    "😐": 0, "😶": 0
}

slang_sentiment = {
    "lmao": 1,
    "omg": -1,
    "meh": 0,
    "wtf": -1,
    "lit": 1,
    "cringe": -1,
    "dope": 1,
    "lol": 1,
    "sucks": -1
}
```

- **VADER** – Base sentiment lexicon
- Custom **emoji** and **slang** sentiment dictionaries

9.2. Scoring Function

```python
# Lexicon scoring function
def lexicon_score(text):
    score = vader.polarity_scores(text)["compound"]

    # Adjust score based on emojis
    for char in text:
        if char in emoji_sentiment:
            score += emoji_sentiment[char]

    # Adjust score based on slang
    for word in text.split():
        if word in slang_sentiment:
            score += slang_sentiment[word]

    # Map to discrete sentiment class
    if score >= 0.3:
        return 2  # Positive
    elif score <= -0.3:
        return 0  # Negative
    else:
        return 1  # Neutral

# Apply lexicon-based scoring to the cleaned text
df["lexicon_label"] = df["clean_text"].apply(lexicon_score)

# View sample results
df[["text", "clean_text", "label", "lexicon_label"]].head(10)
```

A composite sentiment score is calculated from:

1. VADER compound score
2. Emoji sentiment adjustments
3. Slang sentiment adjustments

Scores are mapped to discrete sentiment classes:

- $\geq 0.3 \rightarrow$ Positive (2)
- $\leq -0.3 \rightarrow$ Negative (0)
- Otherwise $\rightarrow$ Neutral (1)

10. Ensemble Model Configuration

10.1. Majority Voting Strategy

```
from sklearn.metrics import classification_report
import numpy as np

# Step 1: Copy BERT Predictions
# Make sure `preds` comes from:
# predictions = trainer.predict(test_dataset)
# preds = predictions.predictions.argmax(-1)

bert_preds = preds  # This comes from BERT model predictions

# Step 2: Copy y_test indices (for matching)
# If y_test is a pandas Series, get index
df_test = df.iloc[y_test.index].copy()

# Step 3: Generate Lexicon Predictions Again (if not done for test only)
df_test["lexicon_label"] = df_test["clean_text"].apply(lexicon_score)

# Step 4: Add BERT Predictions to df_test
df_test["bert_label"] = bert_preds

# Step 5: Majority Vote Logic
def majority_vote(row):
    votes = [row["bert_label"], row["lexicon_label"]]
    return max(set(votes), key=votes.count)

df_test["ensemble_label"] = df_test.apply(majority_vote, axis=1)

# Step 6: Evaluate Hybrid Model
print(" Ensemble (BERT + Lexicon Voting) Performance:\n")
print(classification_report(df_test["label"], df_test["ensemble_label"], target_names=["Negative", "Neutral", "Positive"]))
```

The hybrid model combines BERT and lexicon outputs:

- If both models agree → assign agreed label
- If they differ → choose the majority vote (BERT prioritized in ties)

10.2. Evaluation Metrics

All models are evaluated using:

- Accuracy
- Precision
- Recall
- F1-score (weighted and macro averages)

11. Performance Results

Lexicon Model:

Lexicon Model Evaluation:

	precision	recall	f1-score	support
Negative	0.75	0.18	0.29	1680
Neutral	0.38	0.64	0.48	1294
Positive	0.43	0.59	0.50	1026
accuracy			0.44	4000
macro avg	0.52	0.47	0.42	4000
weighted avg	0.55	0.44	0.40	4000

- Accuracy: 0.44
- F1-score (weighted): 0.40

BERT Model:

Metric	*Value*

eval_loss	0.7239
eval_model_preparation_time	0.0026 seconds
eval_accuracy	0.72625
eval_f1	0.7265
eval_precision	0.7275
eval_recall	0.72625
eval_runtime	18.5359 seconds
eval_samples_per_second	43.159
eval_steps_per_second	5.395

- Accuracy: 0.726
- F1-score (weighted): 0.7265

Ensemble Model:

Ensemble (BERT + Lexicon Voting) Performance:

	precision	recall	f1-score	support
Negative	0.75	0.76	0.75	352
Neutral	0.57	0.75	0.65	238
Positive	0.76	0.48	0.59	210
accuracy			0.68	800
macro avg	0.70	0.66	0.66	800
weighted avg	0.70	0.68	0.68	800

- Accuracy: 0.68
- F1-score (weighted): 0.68

Performance plots compare all models on four metrics using a grouped bar chart.

12. Visualization

A bar chart is generated comparing Accuracy, Precision, Recall, and F1-score for all models. The chart is saved as model_performance_comparison.png.

```python
import matplotlib.pyplot as plt
import pandas as pd

# Data for the three models
data = {
    "Model": ["Lexicon Model", "Ensemble (BERT + Lexicon Voting)", "Transformer-Based BERT Model"],
    "Accuracy": [0.44, 0.68, 0.72625],
    "Precision": [0.55, 0.70, 0.7275],
    "Recall": [0.44, 0.68, 0.72625],
    "F1-Score": [0.40, 0.68, 0.7265]
}

df = pd.DataFrame(data)

# Plotting
plt.figure(figsize=(8, 6))
bar_width = 0.2
x = range(len(df))

plt.bar([i - 1.5*bar_width for i in x], df["Accuracy"], width=bar_width, label="Accuracy")
plt.bar([i - 0.5*bar_width for i in x], df["Precision"], width=bar_width, label="Precision")
plt.bar([i + 0.5*bar_width for i in x], df["Recall"], width=bar_width, label="Recall")
plt.bar([i + 1.5*bar_width for i in x], df["F1-Score"], width=bar_width, label="F1-Score")

plt.xticks(x, df["Model"], rotation=15, ha="right")
plt.ylabel("Score")
plt.ylim(0, 1)
plt.title("Performance Comparison of Models")
plt.legend()
plt.tight_layout()

# Save the figure
plt.savefig("model_performance_comparison.png", dpi=300)
plt.show()
```

References

Rana, M.R.R., Rehman, S.U., Nawaz, A., Ali, T., Imran, A., Alzahrani, A. and Almuhaimeed, A., 2023. Aspect-Based Sentiment Analysis for Social Multimedia: A Hybrid Computational Framework. Comput. Syst. Sci. Eng., 46(2), pp.2415-2428. https://www.researchgate.net/profile/Muhammad-Rizwan-Rashid/publication/368417435_Aspect-Based_Sentiment_Analysis_for_Social_Multimedia_A_Hybrid_Computational_Framework/links/640a1eb9a1b72772e4e55f16/Aspect-Based-Sentiment-Analysis-for-Social-Multimedia-A-Hybrid-Computational-Framework.pdf

Sharma, N. A., Ali, A. S., & Kabir, M. A. 2024. A review of sentiment analysis: Tasks, applications, and deep learning techniques. International Journal of Data Science and Analytics, 17(2), 99–126. https://doi.org/10.1007/s41060-023-00389-z

Sowndarya, C.A., Dahiya, S., Arora, A., Bhardwaj, A., Kumar, M., Ray, M. and Ramasubramanian, V., 2024. Comparative Analysis of Machine Learning and Deep Learning Models for Aspect-based Sentiment Analysis in Education. *Journal of Scientific Research and Reports*, *30*(12), pp.567-576. http://article.ths100.in/id/eprint/1905/1/Dahiya30122024JSRR127894.pdf

Wang, J., Xu, B. and Zu, Y., 2021, July. Deep learning for aspect-based sentiment analysis. In *2021 International conference on machine learning and intelligent systems engineering (MLISE)* (pp. 267-271). IEEE. https://cs224d.stanford.edu/reports/WangBo.pdf