NLP Preprocessing And Text Classification

Course Name: MDM Deep Learning

Lab Title: NLP Techniques for Text Classification

Student Name: Rohit Dahale

Student ID:202201070052

Date of Submission: 16/4/2025

Group Members: Sakshi Aher, Akhilesh Ukey

Objective The objective of this assignment is to implement NLP preprocessing techniques and build a text classification model using machine learning techniques.

Learning Outcomes:

1. Understand and apply NLP preprocessing techniques such as tokenization, stopword removal, stemming, and lemmatization.
2. Implement text vectorization techniques such as TF-IDF and CountVectorizer.
3. Develop a text classification model using a machine learning algorithm.
4. Evaluate the performance of the model using suitable metrics.

```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving all-data.csv to all-data (1).csv
```

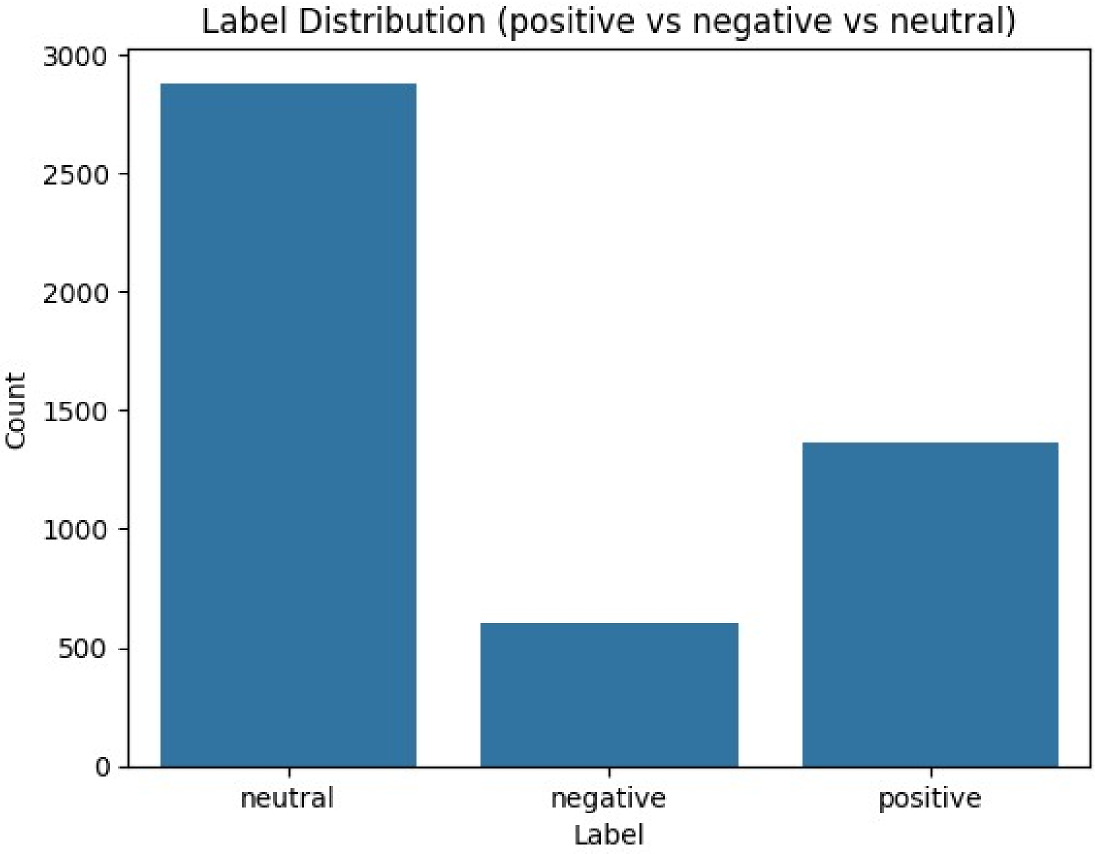
```
import pandas as pd

# Load the CSV with proper encoding
df = pd.read_csv("all-data.csv", encoding='ISO-8859-1', header=None)

# Rename columns
df.columns = ['label', 'text']
df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 4846,\n  \"fields\":
[\n    {\n      \"column\": \"label\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"neutral\",\n          \"negative\",\n
\"positive\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"text\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4838,\n        \"samples\": [\n          \"The
Company serves approximately 3,000 customers in over 100
countries .\",\n          \"On Dec. 1 , Grimaldi acquired 1.5 million
shares and a 50.1-percent stake in Finnlines .\",\n          \"The
extracted filtrates are very high in clarity while the dried filter
cakes meet required transport moisture limits (TMLs)for their ore
grades .\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
# Check for missing values in any column
print("Missing values:\n", df.isnull().sum())

# Print data types of each column
print("\nData Types:\n", df.dtypes)

# Visualize the distribution of labels (ham vs spam)
sns.countplot(x='label', data=df)
plt.title("Label Distribution (positive vs negative vs neutral)")
plt.xlabel("Label")
plt.ylabel("Count")
plt.show()
```

```
Missing values:
 label             0
text              0
processed_text    0
dtype: int64

Data Types:
 label             object
text              object
processed_text    object
dtype: object
```



```python
import nltk

nltk.download('punkt')            # For tokenization
nltk.download('stopwords')        # For removing stopwords
nltk.download('wordnet')          # For lemmatization
nltk.download('omw-1.4')          # WordNet data
nltk.download('punkt_tab')        # <- Specific one mentioned in error
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

```
True
```

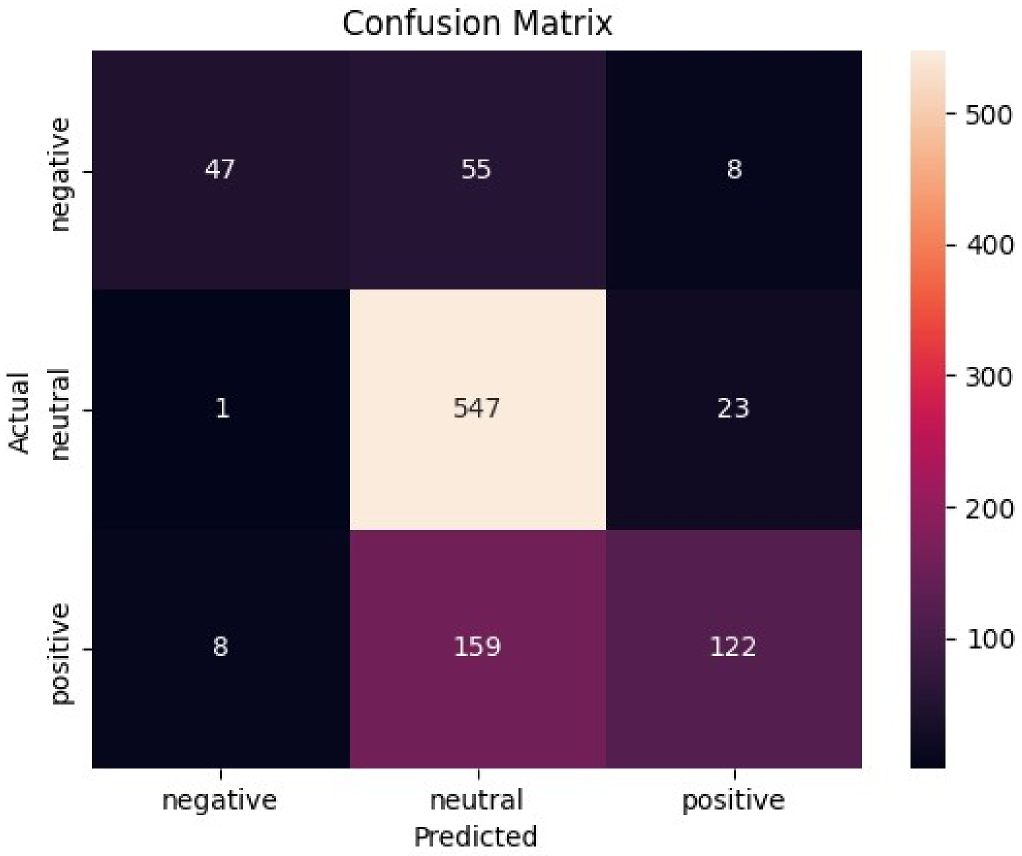
```python
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
import string

stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    tokens = word_tokenize(text)
    tokens = [w for w in tokens if w not in stop_words]
    tokens = [stemmer.stem(lemmatizer.lemmatize(w)) for w in tokens]
    return ' '.join(tokens)

df['processed_text'] = df['text'].apply(preprocess_text)
df[['label', 'processed_text']].head()
```

```
{"summary":"{\n  \"name\": \"df[['label', 'processed_text']]\",\n
\"rows\": 5,\n  \"fields\": [\n    {\n      \"column\": \"label\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"neutral\",\n          \"negative\",\n          \"positive\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"processed_text\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"technopoli plan develop stage area less 100000 squar meter order
host compani work comput technolog telecommun statement said\",\n
\"accord compani updat strategi year 20092012 baswar target longterm
net sale growth rang 20 40 oper profit margin 10 20 net sale\",\n
\"intern electron industri compani elcoteq laid ten employe tallinn
facil contrari earlier layoff compani contract rank offic worker daili
postime report\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```python
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer

# TF-IDF Vectorizer
tfidf = TfidfVectorizer()
X_tfidf = tfidf.fit_transform(df['processed_text'])

# Count Vectorizer
count_vec = CountVectorizer()
X_count = count_vec.fit_transform(df['processed_text'])

# Target labels
y = df['label']
```

```
from sklearn.model_selection import train_test_split

# Using TF-IDF for modeling
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y,
test_size=0.2, random_state=42)

from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred,
average='weighted'))
print("Recall:", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))

# Confusion Matrix
conf_mat = confusion_matrix(y_test, y_pred, labels=model.classes_)
sns.heatmap(conf_mat, annot=True, fmt='d', xticklabels=model.classes_,
yticklabels=model.classes_)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Classification Report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7381443298969073
Precision: 0.7558717224556394
Recall: 0.7381443298969073
F1 Score: 0.7121671058138577
```



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

    negative       0.84      0.43      0.57       110
     neutral       0.72      0.96      0.82       571
    positive       0.80      0.42      0.55       289

    accuracy                           0.74       970
   macro avg       0.79      0.60      0.65       970
weighted avg       0.76      0.74      0.71       970
```

Discussion and Conclusion :

After implementing the text classification pipeline using natural language processing (NLP) techniques on the Financial news dataset, the following model evaluation metrics were observed:

Accuracy: 0.73
Precision: 0.75

Recall: 0.73
F1 Score: 0.71

Declaration

I, Rohit Dahale, confirm that the work submitted in this assignment is my own and has been completed following academic integrity guidelines. The code is uploaded on my GitHub repository account, and the repository link is provided below:

GitHub Repository Link: https://github.com/rohitdahale/NLP-Techniques

Signature: Rohit R. Dahale