NLP Assignment 2 - 22070126093

September 11, 2024

1 NLP Assignment 2 - *IMDB Dataset of 50K Movie Reviews*

Name - Saharsh Mehrotra PRN - 22070126093 AIML B1

IMDB dataset having 50K movie reviews for natural language processing or Text analytics. This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training and 25,000 for testing. So, predict the number of positive and negative reviews using either classification or deep learning algorithms.

1.0.1 *Importing Libraries*

[1]:
```
import pandas as pd
import numpy as np
import re
import spacy
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

# Load spaCy model and download NLTK stopwords
nlp = spacy.load('en_core_web_sm')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[1]:
```
True
```

1.0.2 *Loading the Dataset*

[3]:
```
# Load dataset (replace with actual path)
df = pd.read_csv('/content/IMDB Dataset.csv')

# Remove empty rows and duplicates
df.dropna(subset=['review'], inplace=True)
df.drop_duplicates(subset=['review'], inplace=True)
df.head()
```

[3]:

	review	sentiment
0	One of the other reviewers has mentioned that …	positive
1	A wonderful little production. The…	positive
2	I thought this was a wonderful way to spend ti…	positive
3	Basically there's a family where a little boy …	negative
4	Petter Mattei's "Love in the Time of Money" is…	positive

1.0.3 *Text Cleaning*

[5]:
```
# Contractions and abbreviation handling
contractions = {"isn't": "is not", "can't": "cannot", "he's": "he is", "it's":␣
 ↪"it is", "i'm": "I am", "won't": "will not"}

def clean_text(text):
    text = text.lower()  # Lowercase
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)  #␣
 ↪Remove URLs
    text = re.sub(r'\s+', ' ', text)  # Remove excess whitespaces
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation and symbols
    text = re.sub(r'\d+', '', text)  # Remove numbers
    # Handle contractions
    text = ' '.join([contractions.get(word, word) for word in text.split()])
    return text

# Apply cleaning to the reviews
df['cleaned_review'] = df['review'].apply(lambda x: clean_text(x))
```

1.0.4 *Tokenization, Lemmatization and Stopword Removal*

[6]:
```
# Load stopwords
stop_words = set(stopwords.words('english'))

# Tokenization, lemmatization, and removing stopwords
def preprocess_text(text):
    doc = nlp(text)
    tokens = [token.lemma_ for token in doc if token.text not in stop_words and␣
 ↪not token.is_punct]
```

```
    return ' '.join(tokens)

# Apply lemmatization and stopword removal
df['cleaned_review'] = df['cleaned_review'].apply(preprocess_text)
```

1.0.5 *Label Encoding*

[7]:
```
# Encode sentiment labels
encoder = LabelEncoder()
df['sentiment'] = encoder.fit_transform(df['sentiment'])
```

1.0.6 *Data Splitting*

[8]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['cleaned_review'],␣
 ↪df['sentiment'], test_size=0.2, random_state=42)
```

1.0.7 *Tokenization and Padding*

[9]:
```
# Tokenization and padding
def tokenize_and_pad(train_texts, test_texts, max_words, max_seq_len):
    tokenizer = Tokenizer(num_words=max_words)
    tokenizer.fit_on_texts(train_texts)

    train_sequences = tokenizer.texts_to_sequences(train_texts)
    test_sequences = tokenizer.texts_to_sequences(test_texts)

    max_seq_len = max([len(x) for x in train_sequences])  # Get max sequence␣
 ↪length in dataset
    train_padded = pad_sequences(train_sequences, maxlen=max_seq_len,␣
 ↪padding='post')
    test_padded = pad_sequences(test_sequences, maxlen=max_seq_len,␣
 ↪padding='post')

    return train_padded, test_padded, max_seq_len, tokenizer

# Tokenize and pad for both sets of results
train_padded, test_padded, max_sequence_length, tokenizer =␣
 ↪tokenize_and_pad(X_train, X_test, max_words=10000, max_seq_len=50)
```

1.0.8 *Model Building*

[10]:
```
# Build LSTM model
def build_model(embedding_dim, lstm_units, num_layers, max_words,␣
 ↪max_sequence_length):
```

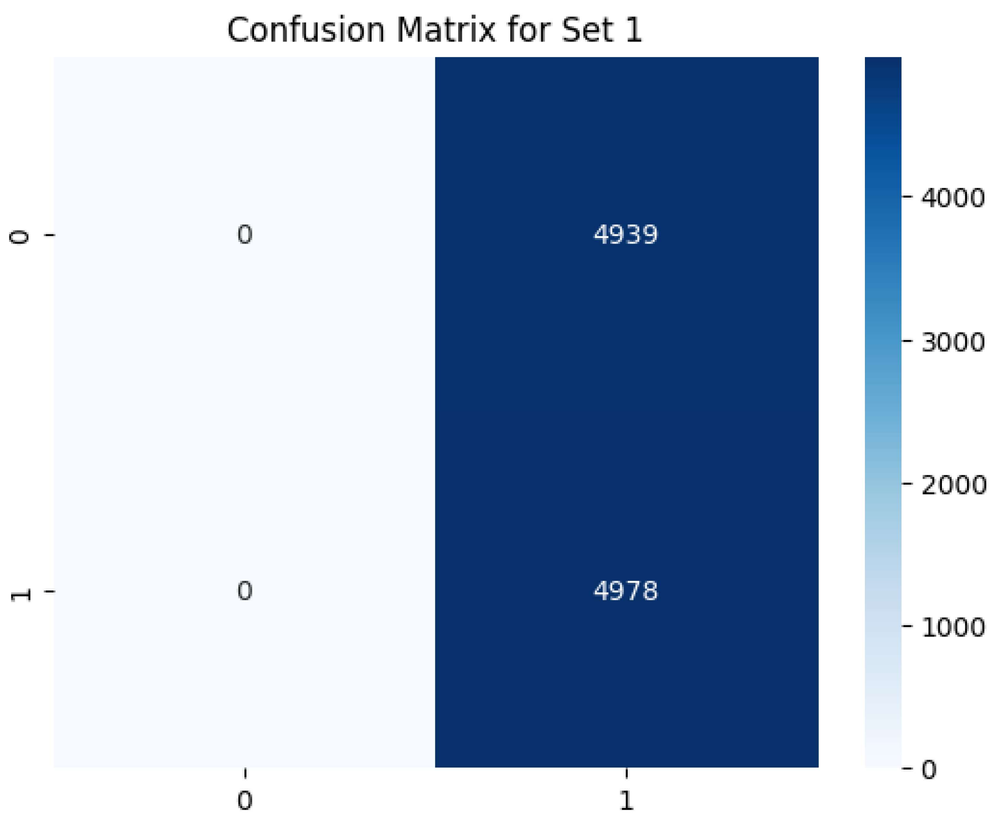
```
    model = Sequential()
    model.add(Embedding(input_dim=max_words, output_dim=embedding_dim,␣
 ↪input_length=max_sequence_length))
    for _ in range(num_layers):
        model.add(LSTM(lstm_units, return_sequences=True))
    model.add(LSTM(lstm_units))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model
```

1.0.9 *Training and Evaluation*

[12]:
```
# Hyperparameters for first and second sets
sets = [
    {'batch_size': 4, 'embedding_dim': 10, 'max_words': 10000, 'lstm_units': 8,␣
 ↪'num_layers': 1},
    {'batch_size': 8, 'embedding_dim': 30, 'max_words': 25000, 'lstm_units':␣
 ↪16, 'num_layers': 2}
]

# Train and evaluate models for both sets
results = {}
for i, params in enumerate(sets):
    model = build_model(params['embedding_dim'], params['lstm_units'],␣
 ↪params['num_layers'], params['max_words'], max_sequence_length)
    model.fit(train_padded, y_train, epochs=1, batch_size=params['batch_size'],␣
 ↪validation_split=0.2)

    y_pred = (model.predict(test_padded) > 0.5).astype('int32')

    print(f"Classification Report for Set {i+1}:")
    print(classification_report(y_test, y_pred))

    # Confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix for Set {i+1}')
    plt.show()
```

```
7933/7933              5211s
656ms/step - accuracy: 0.4951 - loss: 0.6935 - val_accuracy: 0.5026 - val_loss:
0.6932
310/310             45s 143ms/step
Classification Report for Set 1:
              precision    recall  f1-score   support
```

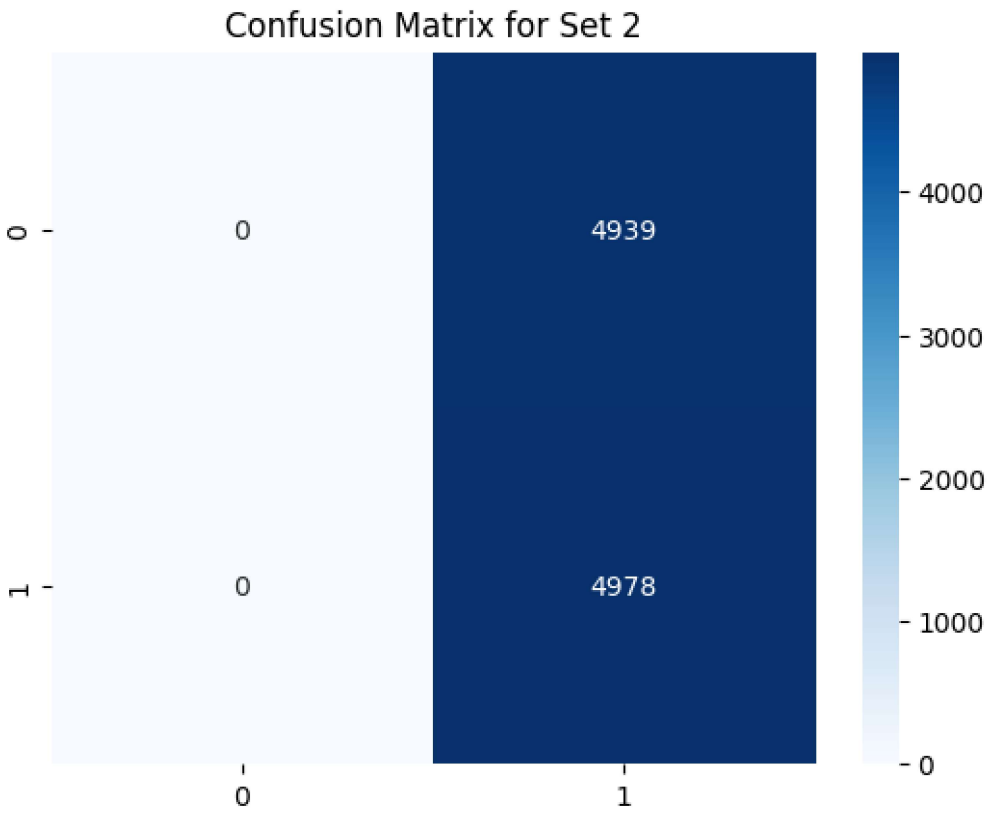
```
           0       0.00      0.00      0.00      4939
           1       0.50      1.00      0.67      4978

    accuracy                           0.50      9917
   macro avg       0.25      0.50      0.33      9917
weighted avg       0.25      0.50      0.34      9917

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90:
UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
```

```
3967/3967            5119s 1s/step -
accuracy: 0.4984 - loss: 0.6936 - val_accuracy: 0.5026 - val_loss: 0.6931
310/310            73s 236ms/step
Classification Report for Set 2:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      4939
           1       0.50      1.00      0.67      4978

    accuracy                           0.50      9917
   macro avg       0.25      0.50      0.33      9917
weighted avg       0.25      0.50      0.34      9917
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



[13]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[14]:
```
# Convert notebook to PDF
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
!jupyter nbconvert --to PDF "/content/drive/MyDrive/Colab Notebooks/NLP␣
↪Assignment 2 - 22070126093.ipynb"
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
  fonts-urw-base35 libapache-pom-java libcmark-gfm-extensions0.29.0.gfm.3
libcmark-gfm0.29.0.gfm.3
  libcommons-logging-java libcommons-parent-java libfontbox-java libfontenc1
libgs9 libgs9-common
```