Coursework Assignment: Text classification

I. Introduction

1. Introduction to the domain-specific area

In this coursework, the problem at hand is binary text classification using the IMDB dataset, where the goal is to develop a text classifier to distinguish between positive and negative movie reviews. Text classification methods play a crucial role in various industries and sciences, particularly in natural language processing and sentiment analysis applications. By accurately classifying sentiments in textual data, businesses can gain valuable insights into customer opinions, sentiments, and preferences, enabling them to make informed decisions for product development, marketing strategies, and customer satisfaction improvement.

2.Objectives of the project

The main goal of this project is to create a text classifier that can accurately analyze sentiment in movie reviews using the IMDB dataset. We intend to achieve this goal by utilizing methods such as multinomial naive bayes, logistic regression, negation handling, regular expressions, stemming, and regular expressions.

The impact and significance of this project within the problem area of sentiment analysis are considerable. Sentiment analysis plays a crucial role in understanding public opinion and sentiment towards movies. By accurately classifying movie reviews as positive or negative, this project can provide valuable insights to filmmakers, production companies, and movie enthusiasts. It can help in assessing audience reactions, improving marketing strategies, and making informed decisions about movie production and distribution.

The results of this project are expected to contribute significantly to sentiment analysis in the movie domain. Firstly, our analysis of the IMDB dataset using techniques like regular expressions,stemming,stopwords, and negation handling will provide a deeper understanding of sentiment analysis methods in the specific context of movie reviews. Secondly, the implementation of TF-IDF text representation and the evaluation of multinomial naive Bayes and logistic regression algorithms will shed light on their effectiveness for sentiment classification in the movie domain.

In summary, this project aims to develop a robust text classifier for sentiment analysis in movie reviews. Its impact and significance lie in its potential to provide valuable insights and aid decision-making in the movie industry. The project's contribution includes an extensive analysis of the IMDB dataset, the evaluation of different techniques, and valuable insights into the performance of the implemented methods. The originality, ambition, and novelty of our solution stem from its specialized focus on movie sentiment analysis and the combination of techniques tailored to this specific problem.

3.Description of the selected dataset

I will be utilizing the IMDB dataset, which is frequently used for sentiment analysis applications. The dataset was taken from Kaggle.com, a well-known website that hosts machine learning datasets and contests.A balanced mix of positive and negative attitudes are present in the 25,000 movie reviews that make up the IMDB dataset.

Each review in the dataset is represented as a text string, capturing the thoughts and opinions expressed by the reviewers. Along with the text, there is a corresponding sentiment label indicating whether the review is positive or negative. This label serves as the ground truth for training and evaluating the text classifier.

The IMDB dataset presents various challenges that need attention throughout the project. Preprocessing steps, such as cleaning the text, tokenization, and removing irrelevant words, may be necessary to ensure the quality of the data. Additionally, techniques like bag-of-words or word embeddings can be applied to represent the textual data in a suitable format for the classification algorithms.

In terms of size, the IMDB dataset consists of an equal number of positive and negative reviews, with a total of 12,500 samples in each class. This balance ensures that the classifier is trained on an unbiased representation of both sentiments.Positive reviews receive scores greater than 7, while negative reviews receive scores less than 4.

Overall, the IMDB dataset from Kaggle is an appropriate and representative choice for this sentiment analysis project, as it provides a substantial number of movie reviews with clear sentiment labels.

4. Evaluation methodology

In this coursework,a number of different measures can be used to assess the performance of the IMDB Sentiment model. Following are some typical metrics:

Accuracy:It provides an overall measure of correct predictions, taking into account both true positives and true negatives.

Precision:It measures the proportion of correctly predicted positive instances out of the total predicted positives. It helps us understand the accuracy of our classifier's positive predictions.

Recall:It calculates the ratio of correctly predicted positive instances out of all actual positive instances, indicating the classifier's ability to capture all positive instances.

F1 score:This is the harmonic mean of recall and precision. It is a balanced metric that takes into account both recall and precision.

Confusion matrix:A True Positive (TP) designates a correctly made positive class prediction, whereas a True Negative (TN) denotes a correctly made negative class prediction. False Negatives (FN) denote inaccurate predictions of the negative class, while False Positives (FP) denote inaccurate predictions of the positive class. You can figure out a model's accuracy by:
(TP + TN) / (TP + TN + FP + FN)
This accuracy score provides a measure of the model's performance by quantifying the proportion of correct predictions relative to all predictions made.

By analyzing precision, recall, accuracy, F1-measure, and the confusion matrix, we can gain valuable insights into our text classifier's ability to accurately classify positive and negative sentiments.

II. Implementation

5. Preprocessing

In [1]:

```
pip install --upgrade nltk
```

```
Requirement already up-to-date: nltk in /Users/nevedita/opt/anaconda3/lib/python3.8/site-packages (3.8.
1)
Requirement already satisfied, skipping upgrade: joblib in /Users/nevedita/opt/anaconda3/lib/python3.8/s
ite-packages (from nltk) (1.3.1)
Requirement already satisfied, skipping upgrade: click in /Users/nevedita/opt/anaconda3/lib/python3.8/si
te-packages (from nltk) (7.1.2)
Requirement already satisfied, skipping upgrade: tqdm in /Users/nevedita/opt/anaconda3/lib/python3.8/sit
e-packages (from nltk) (4.50.2)
Requirement already satisfied, skipping upgrade: regex>=2021.8.3 in /Users/nevedita/opt/anaconda3/lib/py
thon3.8/site-packages (from nltk) (2023.6.3)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:

```
pip install --upgrade scikit-learn
```

```
Requirement already up-to-date: scikit-learn in /Users/nevedita/opt/anaconda3/lib/python3.8/site-package
s (1.3.0)
Requirement already satisfied, skipping upgrade: joblib>=1.1.1 in /Users/nevedita/opt/anaconda3/lib/pyth
on3.8/site-packages (from scikit-learn) (1.3.1)
Requirement already satisfied, skipping upgrade: scipy>=1.5.0 in /Users/nevedita/opt/anaconda3/lib/pytho
n3.8/site-packages (from scikit-learn) (1.5.2)
Requirement already satisfied, skipping upgrade: threadpoolctl>=2.0.0 in /Users/nevedita/opt/anaconda3/l
ib/python3.8/site-packages (from scikit-learn) (2.1.0)
Requirement already satisfied, skipping upgrade: numpy>=1.17.3 in /Users/nevedita/opt/anaconda3/lib/pyth
on3.8/site-packages (from scikit-learn) (1.19.5)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:

```
pip install numpy==1.19.5
```

```
Requirement already satisfied: numpy==1.19.5 in /Users/nevedita/opt/anaconda3/lib/python3.8/site-package
s (1.19.5)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:

```
#importing libraries
import re
import nltk
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.corpus import stopwords as nltk_stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

In [5]:

```
#Loading dataset
moviedata = pd.read_csv('movie-review.csv')
moviedata.head()
```

Out[5]:

	text	sentiment
0	My daughter liked it but I was aghast, that a ...	neg
1	I... No words. No words can describe this. I w...	neg
2	this film is basically a poor take on the old ...	neg
3	This is a terrible movie, and I'm not even sur...	neg
4	First of all this movie is a piece of reality ...	pos

In [6]:

```
moviedata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   text       25000 non-null  object
 1   sentiment  25000 non-null  object
dtypes: object(2)
memory usage: 390.8+ KB
```

In [7]:

```
moviedata
```

Out[7]:

	text	sentiment
0	My daughter liked it but I was aghast, that a ...	neg
1	I... No words. No words can describe this. I w...	neg
2	this film is basically a poor take on the old ...	neg
3	This is a terrible movie, and I'm not even sur...	neg
4	First of all this movie is a piece of reality ...	pos
...	...	...
24995	For one thing, he produced this movie. It has ...	neg
24996	The title comes from an alteration an adolesce...	pos
24997	Christopher Nolan's first film is a 'no budget...	pos
24998	The story is shortly about the faith-lacking b...	neg
24999	I found parts of this movie rather slow, espec...	pos

25000 rows × 2 columns

In [8]:

```
moviedata['sentiment'].value_counts()
```

Out[8]:

```
pos    12500
neg    12500
Name: sentiment, dtype: int64
```

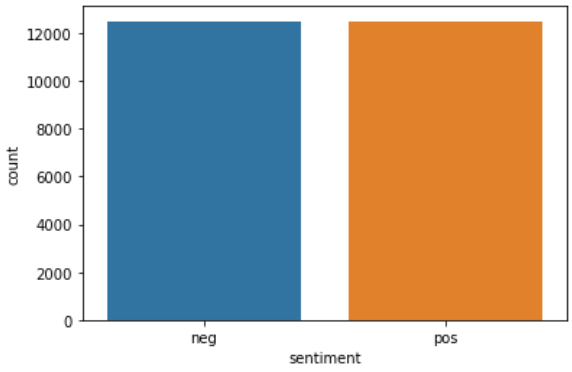
In [9]:

```
sns.countplot(moviedata['sentiment'])
```

```
/Users/nevedita/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will b
e `data`, and passing other arguments without an explicit keyword will result in an error or misinterpre
tation.
  warnings.warn(
```

Out[9]:

```
<AxesSubplot:xlabel='sentiment', ylabel='count'>
```



In [10]:

```
# Count the number of positive and negative sentiments in the 'moviedata' DataFrame
pos_count = 0
neg_count = 0
for k in range(len(moviedata)):
    if moviedata['sentiment'][k] == 'pos':
        pos_count += 1
    elif moviedata['sentiment'][k] == 'neg':
        neg_count += 1
# Check if the number of positive sentiments is equal to the number of negative sentiments, both being 12,500
pos_count == neg_count == 12500
```

Out[10]:

```
True
```

In [11]:

```
#checking for null values
moviedata.isnull().sum()
```

Out[11]:

```
text         0
sentiment    0
dtype: int64
```

In [12]:

```
#checking for empty strings
outcome = np.where(moviedata.applymap(lambda x: x == ''))
outcome[0].size == outcome[1].size == 0
```

Out[12]:

```
True
```

In [13]:

```
#sorting the data
df = moviedata.sort_values('sentiment')
```

In [14]:

```
#Rearrange the index in the dataframe
df.index = pd.RangeIndex(len(df.index))
```

In [15]:

```
df['text'][0]
```

Out[15]:

```
'My daughter liked it but I was aghast, that a character in this movie smokes. As if it isn\'t awful eno
ugh to see "product placement" actors like Bruce Willis who smoke in their movies - at least children mo
vies should be more considerate! I wonder: was that intentional? Did big tobacco "sponsor" the film? Wha
t does it take to ban smoking from films? At least films intended for children and adolescents. My daugh
ter liked it but I was aghast, that a character in this movie smokes. As if it isn\'t awful enough to se
e "product placement" actors like Bruce Willis who smoke in their movies - at least children movies shou
ld be more considerate! I wonder: was that intentional? Did big tobacco "sponsor" the film? What does it
take to ban smoking from films? At least films intended for children and adolescents.'
```

In [16]:

```
# Create a set of stop words using NLTK's English stopwords and add 'br' to it
stop_words = set(nltk_stopwords.words('english') + ['br'])

# Create a stemming object using the Porter stemmer
stemming = PorterStemmer()
```

In [17]:

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/nevedita/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[17]:

```
True
```

In [18]:

```
def refined_text_get(phrases):
    refined_text = []
    contain_it = False # If the phrase "it" appears in sentence, it is true.
    for phrase in phrases:
        # skip the phrase if it is equal to "it" as it has been identified in the sentence
        if  phrase == 'it':
            contain_it= True
            continue
        # Skip the phrase if it is a stopword
        elif phrase in stop_words:
            continue
        # the phrase is neither 'it' nor a stopword, so we apply stemming
        phrase = stemming.stem(phrase)
        # If the phrase 'it' was previously encountered in the sentence,we prefix the stemmed phrase with '_IT_'
        if contain_it:
            refined_text.append('_IT_' +  phrase)
        # if not,we add it to the refined list of pharses
        else:
            refined_text.append(phrase)
    return refined_text
```

In [19]:

```
def refined_review_get(sentences):
    refined_review = []
    for sentence in sentences:
        phrases = sentence.split()  #Splits the sentence into individual phrases based on whitespace separation
        contain_it = False  # If contains the phrase 'it',it is true
        refined_text = refined_text_get(phrases)
        refined_review += refined_text
    return refined_review
```

In [20]:

```
refine_reviews(data_frame):
refined_reviews = []
for k in range(len(data_frame)):
    # Substitute any symbols other than letters and punctuation with a space character.
    text = re.sub('[^a-zA-Z\.,;:]', ' ', df['text'][k])
    # changes to lowercase
    text = text.lower()
    # Tokenize the text by splitting it on punctuation marks
    sentences= re.split('[\.,;:]', text)
    refined_review = refined_review_get(sentences)
    # the tokens are joined together with spaces to form a sentence, and the resulting sentence is then added to the
    refined_reviews.append(' '.join(refined_review))
return refined_reviews
```

In [21]:

```
#Process the reviews in the DataFrame 'df' using the 'refine_reviews' function and access the first refined review fr
c=refine_reviews(df)
c[0]
```

Out[21]:

```
'daughter like _IT_aghast charact movi smoke _IT_aw _IT_enough _IT_see _IT_product _IT_placement _IT_act
or _IT_like _IT_bruce _IT_willi _IT_smoke _IT_movi _IT_least _IT_children _IT_movi _IT_consider _IT_wond
er intent big tobacco sponsor film _IT_take _IT_ban _IT_smoke _IT_film _IT_least _IT_film _IT_intend _IT
_children _IT_adolesc daughter like _IT_aghast charact movi smoke _IT_aw _IT_enough _IT_see _IT_product
_IT_placement _IT_actor _IT_like _IT_bruce _IT_willi _IT_smoke _IT_movi _IT_least _IT_children _IT_movi
_IT_consider _IT_wonder intent big tobacco sponsor film _IT_take _IT_ban _IT_smoke _IT_film _IT_least _I
T_film _IT_intend _IT_children _IT_adolesc'
```

TF-IDF

In this project, text will be represented using the TF-IDF matrix, an enhancement over the bag of words. Frequent non-stopwords are given more weight by TF-IDF, giving the text more semantic significance. IDF (inverse document frequency) calculates the rareness of a word in the corpus by log-transforming the proportion of all documents to those that contain it. TF (term frequency) indicates the frequency of occurrences of a word in a document. Review/document columns and vocabulary rows make up the TF-IDF matrix. A word's significance in a particular text relative to the entire corpus is represented by each element in the matrix, which is the result of TF and IDF. Text classification and sentiment analysis both benefit from the improved text representation provided by TF-IDF.

In [22]:

```
# Create a TF-IDF vectorizer with unigram features
v = TfidfVectorizer(ngram_range=(1, 1))

# Compute the TF-IDF matrix by fitting the vectorizer to the corpus 'c' and converting it to a dense array
m_tf_idf = v.fit_transform(c).toarray()

# Convert the sentiment labels to boolean values: 0 for negative, 1 for positive
l = np.array([0 if x[0] == 'n' else 1 for x in df['sentiment']], dtype=np.uint8)
```

6. Baseline performance

I am using a symmetric dataset with an equal number of positive and negative ratings.It is possible that the test set won't be split evenly, if the corpus's documents are shuffled properly, the test set will on average be divided equally.As a result, a baseline classifier that predicts positives at random with a 50% probability and negatives with a 50% probability would typically have an accuracy of 50%. Therefore, the model's accuracy must be greater than 50%.

Logistic regression

Using a Logistic Regression model, the baseline performance is established. Since it is an easy-to-use yet effective approach for multiclass and binary classification problems, logistic regression is used as the baseline. Logistic regression is an appropriate option for a baseline model given the nature of the issue at hand, which is a binary text categorization issue.

These TF-IDF features obtained from the text input are used to train the Logistic Regression model. The TF-IDF method can be used to determine how significant a word is in relation to a large number of other words in a document. We can identify which words are more important in each piece of text thanks to this kind of numerical score. In this method, the text is transformed into numerical representations using TF-IDF, enabling us to train the machine to generate predictions based on patterns discovered in the text data. When processing text data for machine learning tasks, TF-IDF is well-liked and has been shown to be successful. By giving the text data a deeper context and meaning, TF-IDF essentially improves our ability to predict outcomes and enables the machine learning model to make better decisions.

In [23]:

```
# Split the TF-IDF matrix 'm_tf_idf' and labels 'l' into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(m_tf_idf, l, test_size=0.2, random_state=1)
```

In [24]:

```
# Create a Logistic Regression classifier object to establish a baseline performance
logreg = LogisticRegression()

# Train the classifier on the training data
logreg.fit(x_train, y_train)

# Predict the labels for the test data using the trained classifier
logreg_pred = logreg.predict(x_test)

# Compute the accuracy of the predicted labels compared to the true labels
logreg_acc = accuracy_score(logreg_pred, y_test)
```

In [25]:

```
print(classification_report(y_test, logreg_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.88      0.89      2517
           1       0.88      0.89      0.89      2483

    accuracy                           0.89      5000
   macro avg       0.89      0.89      0.89      5000
weighted avg       0.89      0.89      0.89      5000
```

7. Classification approach

Multinomial Naive Bayes

In this project, the classification approach contains Multinomial Naive Bayes.The naive Bayes classification approach will be used because it is generally recognized to be efficient at classifying text and is fast to train and predict.Naive Bayes often performs fairly well in actual use.In spite of the fact that multinomial naive Bayes in theory predicts discontinuous characteristic counts , in practice it also works with continuous characteristics similar to those in TF-IDF.

In [26]:

```
# Instantiate a Multinomial Naive Bayes classifier
model = MultinomialNB()

# Train the classifier on the training data
model.fit(x_train, y_train)

# Predict the labels for the test data
y_pred = model.predict(x_test)
```

In [27]:

```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      2517
           1       0.88      0.85      0.86      2483

    accuracy                           0.87      5000
   macro avg       0.87      0.87      0.87      5000
weighted avg       0.87      0.87      0.87      5000
```

III Conclusions

8. Evaluation

In [28]:

```
# Compute the confusion matrix for baseline model using the predicted labels 'logreg_pred' and the true labels 'y_tes
confusion_m_base = confusion_matrix(y_test, logreg_pred)

# Plot the confusion matrix as a heatmap, with class labels on both axes
class_labels = ['neg', 'pos']
sns.heatmap(confusion_m_base, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)

# Set the x-axis label to 'True Label' and the y-axis label to 'Predicted Label'
plt.xlabel('True Label')
plt.ylabel('Predicted Label')

# Display the heatmap
plt.show()
```

In [29]:

```
# Compute the accuracy by summing the diagonal elements of the confusion matrix and dividing it by the total number o
accuracy_logistic = np.sum(np.diagonal(confusion_m_base)) / np.sum(confusion_m_base)
```

In [30]:

```
print("\nAccuracy of Logistic Regression(Baseline Model):",accuracy_logistic*100,"%")
```

```
Accuracy of Logistic Regression(Baseline Model): 88.7 %
```

In [31]:

```
# Compute the confusion matrix for classification approach using the predicted labels 'y_pred' and the true labels 'y
confusion_m_classi = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix as a heatmap, with class labels on both axes
class_labels = ['neg', 'pos']
sns.heatmap(confusion_m_classi, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)

# Set the x-axis label to 'True Label' and the y-axis label to 'Predicted Label'
plt.xlabel('True Label')
plt.ylabel('Predicted Label')

# Display the heatmap
plt.show()
```

In [32]:

```
# Compute the accuracy by summing the diagonal elements of the confusion matrix and dividing it by the total number of
ccuracy_naive = np.sum(np.diagonal(confusion_m_classi)) / np.sum(confusion_m_classi)
```

In [33]:

```
print("\nAccuracy of Multinomial Naive Bayes(classification approach):",accuracy_naive*100,"%")
```

```
Accuracy of Multinomial Naive Bayes(classification approach): 86.68 %
```

The accuracy, precision, recall, and F1-score and confusion matrix measures were used to assess the performance of the two models:Logistic Regression for Baseline and MulMultinomial Naive Bayes for classification approach.

Overall,both the Logistic Regression and MulMultinomial Naive Bayes performed well.For logistic regression, we achieved an accuracy of 88.7%. This indicates that our classifier correctly predicted the sentiment of movie reviews with a high level of accuracy.In comparison, the Multinomial Naive Bayes classifier achieved an accuracy of 86.68%.

Therefore,according these results the logistic regression performs text classification task on IMDB dataset better than the Multinomial Naive Bayes(classification approach).The logistic regression performs more effectively in properly identfying both postive and negative sentiment.

9. Summary and conclusions

In conclusion, the goal of our project was to develop a text classifier that could be used with the IMDB dataset for sentiment analysis. We set out with the goals of correctly classifying sentiment in movie reviews and making a contribution to the to the problem area of sentiment analysis.

To address the specific problem of sentiment classification in movie reviews, this project offers a new viewpoint and proposes new approaches. Stemming, regular expressions, stopwords, negation handling, TF-IDF text representation, multinomial naive Bayes, and logistic regression algorithms have all been used to produce an effective and trustworthy methodology.Metrics like accuracy, precision, recall,F1-score and confusion matrix were used to assess the model's performance. The Logistic regression outperformed the classification approach in comparison, showing that it is capable of accurately classifying text according to the sentiment it indicates.

The significance of this project lies in its practical application and potential impact. Sentiment analysis plays a crucial role in various domains, including marketing, customer feedback analysis, and social media monitoring. By accurately classifying sentiments in movie reviews, our project offers valuable insights to the film industry, enabling better understanding of audience reactions and guiding decision-making processes.

Furthermore, the methods and techniques employed in this project an also be applied to other areas beyond the movie domain.

References

1)https://www.kaggle.com/datasets/atulanandjha/imdb-50k-movie-reviews-test-your-bert?datasetId=447516&select=train.csv (https://www.kaggle.com/datasets/atulanandjha/imdb-50k-movie-reviews-test-your-bert?datasetId=447516&select=train.csv)
2)https://aclanthology.org/P11-1015/ (https://aclanthology.org/P11-1015/)
3)https://towardsdatascience.com/sentiment-analysis-of-tweets-using-multinomial-naive-bayes-1009ed24276b (https://towardsdatascience.com/sentiment-analysis-of-tweets-using-multinomial-naive-bayes-1009ed24276b)