**SOFTWARE ENGINEERING, UNVERSITY OF KELANIYA**

**BIG DATA ANALYTICS**

**SENG 42293**

**IMDB Movie Reviews**

**Sentiment Analysis**

**Report**

**NAME: S. Sivanujan**

**REG. NO:**

**INDEX NO: SE/2017/041**

**EMAIL: ssivanujan1998@gmail.com**

**DATE OF SUBMISSION: 17/05/2023**

**Introduction**

This Assignment has been helped to understand the importance of Sentiment Analysis. Sentiment analysis, also referred to as opinion mining, is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text. This is a popular way for organizations to determine and categorize opinions about a product, service or idea. Sentiment analysis involves the use of data mining, machine learning (ML), artificial intelligence and computational linguistics to mine text for sentiment and subjective information such as whether it is expressing positive, negative or neutral feelings.

This assignment contains the dataset that describes the IMDb movie reviews. The review and sentiment are the selected columns for this analysis. IMDb (Internet Movie Database) is a popular online database that provides information about movies, TV shows, actors, and other industry-related content. One of the features of IMDb is its movie review section, where users can rate and write reviews for films. IMDb movie reviews serve as a platform for movie enthusiasts to share their experiences and opinions with others. Users can read these reviews to get an idea of how a particular movie is perceived by the general audience. The reviews are typically accompanied by the user's rating for the movie, which helps in understanding the overall reception of the film. It's important to note that IMDb movie reviews are subjective and can vary greatly in quality and perspective. Different users may have different tastes and preferences, so it's advisable to read multiple reviews and consider a range of opinions before forming your own judgment about a movie.

Sentiment analysis for IMDb movie reviews is helps in gauging how the general audience perceives a movie. Also, for filmmakers, studios, and distributors, sentiment analysis can help monitor the reputation of their movies. It can assist in understanding audience preferences, identifying trends, and making informed decisions about movie production, marketing, and distribution strategies. Sentiment analysis can be employed in recommendation systems to suggest movies to users based on their preferences. Finally, Sentiment analysis of IMDb movie reviews is also of interest to researchers studying various aspects of cinema, such as film criticism, audience reception, and cultural analysis.

According to the above benefits, sentiment analysis of IMDb movie reviews aims to extract and analyze the sentiment conveyed in the reviews to gain insights into audience opinions, assist decision-making processes, and provide valuable data for various applications in the film industry.

**Data and Description**

Overall given data set contain 50000 reviews. Also have equal numbers of positive and negative reviews for the analysis.

**Data Preparation**

Data cleaning in sentiment analysis is the process of removing redundant and incorrect values in data that is meant for analysis. This is a necessary step in the sentiment analysis process, whatever the business requirement may be - whether customer experience analysis, employee satisfaction analytics, or brand experience insights. Removing all the unnecessary data items that do not belong in your dataset is an essential part of sentiment analysis data preparation, without which the insights you receive will be inaccurate and cannot be relied on.

Initially for this sentiment Analysis I have used some basic cleaning steps. So, first I have removed HTML tags. Because HTML tags do not contribute to the sentiment of the text and can potentially introduce noise. Then I removed non-alphabetic characters. non-alphabetic characters such as numbers, punctuation marks, and special symbols may not provide meaningful sentiment information. Removing them helps to focus on the sentiment-bearing words. Finally, I converted text to lowercase. By converting all the text to lowercase, you ensure consistency in the data and avoid duplicating words with different cases. This step helps in standardizing the text, making it easier for subsequent analysis. Some other cleaning steps (Tokenized) have been covered in the advance cleaning step in the final part of this report. I tried to prove that the accuracy of the sentiment analysis depends on the data cleaning process.

Next important steps of the data preparation are create the bag-of-words and TF-IDF representations. The bag-of-words model represents text as a vector of word occurrences.

In this step, you can use the CountVectorizer class from sklearn.feature\_extraction.text to create a BoW representation. So first, Initialize the CountVectorizer object, for that Set any desired parameters, such as removing stop words, adjusting n-gram range, or specifying the maximum vocabulary size. Then Fit and transform the text data, for that Apply the fit\_transform method of the CountVectorizer object to your tokenized text data. This step generates the BoW representation, where each row corresponds to a document, and each column represents a unique word.

Here I have added the code Snippet for above case:

#import count vectorizer

from sklearn.feature\_extraction.text import CountVectorizer

vectorizer\_bow = CountVectorizer()

X\_bow = vectorizer\_bow.fit\_transform(data['review'])

Same as, TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical representation that assigns weights to words based on their frequency in a document and their importance in the corpus. TF-IDF helps to highlight words that are both frequent within a document and relatively rare in the entire corpus. So first, Initialize the TfidfVectorizer object, for that Set parameters similar to the CountVectorizer step, such as removing stop words, adjusting n-gram range, or specifying the maximum vocabulary size. Then Fit and transform the tokenized text data, for that Use the fit\_transform method of the TfidfVectorizer object on your tokenized text data. This step generates the TF-IDF representation, where each row represents a document, and each column corresponds to a unique word.

Here I have added the code Snippet for above case:

#import TfidfVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

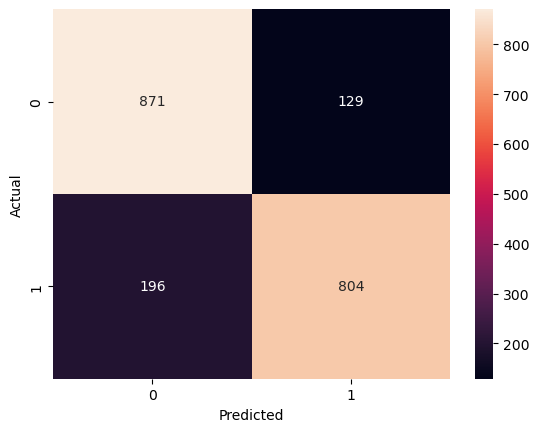
vectorizer\_tfidf = TfidfVectorizer()

X\_tfidf = vectorizer\_tfidf.fit\_transform(data['review'])

For the Model building Split the data into training and testing sets, using 80% of the data for training and the rest for testing. The following section will Evaluate the different types of model classifiers with the appropriate the Messurements.

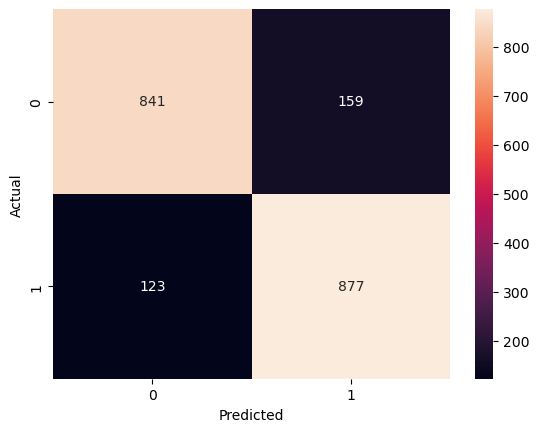
**Classification Models**

In this section, I evaluate the performance of my model using the test set. Compare and contrast the performance of the TF-IDF and bag-of-words methods. For this evaluation, I have used the confusion matrix and the classification report to compare the accuracy, precision, recall, and F1-score of each model.

Confusion Matrix - Naive Bayes with Bag-of-Words

A picture containing text, screenshot, diagram, design

Description automatically generatedConfusion Matrix - Naive Bayes with TF-IDF

Confusion Matrix - Logistic Regression with Bag-of-Words

A picture containing text, screenshot, diagram, design

Description automatically generatedConfusion Matrix - Logistic Regression with TF-IDF

Table 1: Summary of Performance Metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Naive Bayes with Bag-of-Words** | 0.8375 | 0.8390 | 0.8375 | 0.8373 |
| **Naive Bayes with TF-IDF** | 0.8460 | 0.8510 | 0.8460 | 0.8454 |
| **Logistic Regression with Bag-of-Words** | 0.8590 | 0.8595 | 0.8590 | 0.8590 |
| **Logistic Regression with TF-IDF** | 0.8715 | 0.8724 | 0.8715 | 0.8714 |

Figure 1: Performance Metrics Line Chart

According to the above results (Table1, Figure1), Overall, the logistic regression models consistently outperformed the Naive Bayes models in terms of accuracy, precision, recall, and F1 score. Additionally, the TF-IDF representation generally provided better performance than the bag-of-word representation. Comparing the models, logistic regression with TF-IDF achieved the highest accuracy, precision, recall, and F1 score among all the models. This indicates that it performed the best overall in predicting sentiment from the IMDb movie reviews dataset. The logistic regression model with BoW also performed well, but slightly lower than the TF-IDF variant. The TF-IDF representation improved the performance of both the Naive Bayes and logistic regression models compared to the bag-of-words representation. TF-IDF captures the importance of words in the documents, allowing the models to better distinguish between meaningful and less informative words. This leads to better sentiment analysis performance.

It's important to note that these results are specific to the IMDb movie reviews dataset and may not necessarily generalize to other sentiment analysis tasks or datasets. Factors such as the size and quality of the dataset, as well as the specific characteristics of the reviews, can influence the performance of the models. In conclusion, based on the provided results, the logistic regression model with TF-IDF achieved the highest performance in terms of accuracy, precision, recall, and F1 score, demonstrating its effectiveness for sentiment analysis on IMDb movie reviews.

**Conclusions and Discussion**

The Logistic Regression model with TF-IDF achieved the highest accuracy of 0.8715 among all the models evaluated. Accuracy is an important metric for sentiment analysis as it measures the overall correctness of the sentiment predictions. The precision and recall values for the logistic regression model with TF-IDF are relatively high and balanced. Precision measures the proportion of correctly predicted positive sentiments, while recall measures the proportion of correctly predicted positive sentiments out of all the actual positive sentiments. Having balanced precision and recall indicates that the model performs well in correctly classifying positive sentiments without sacrificing its ability to detect all positive sentiments.

The strengths of this model are achieving high accuracy, precision, recall, and F1 score values, indicating its effectiveness in classifying sentiment in IMDb movie reviews, and the use of TF-IDF helps capture the importance of words and can improve the model's understanding of sentiment-specific language. The main limitation is that the evaluation metrics were calculated on a specific dataset. It's important to assess the model's performance on different datasets to ensure its generalizability. There are some issues with the data and models used, such as the fact that the performance of sentiment analysis models can be affected by dataset bias. If the training data is imbalanced or contains biases, the model may struggle to generalize well to unseen data or different domains. And Sentiment analysis models often rely on individual words or word frequencies without considering the context in which they are used. This can lead to misinterpretations of sentiment, especially in cases where sarcasm or subtle language is involved.

To improve the model performance here I suggested some methods with the appropriate evidence. So, first Strategy is using N-grams, instead of considering only individual words, you can also include n-grams (sequences of consecutive words) as features. This can capture more contextual information and improve the model's understanding of the text. And the second strategy is using some advanced text pre-processing, it improves the text cleaning process by implementing techniques like lemmatization or stemming to reduce words to their base or root form. This can help in reducing feature space and capturing the core meaning of words.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Naive Bayes with Bag-of-Words** | 0.8365 | 0.8377 | 0.8365 | 0.8364 |
| **Naive Bayes with TF-IDF** | 0.8530 | 0.8553 | 0.8530 | 0.8528 |
| **Logistic Regression with Bag-of-Words** | 0.8590 | 0.8595 | 0.8590 | 0.8590 |
| **Logistic Regression with TF-IDF** | 0.8715 | 0.8724 | 0.8715 | 0.8714 |

Table 2: Summary of Performance Metrics after Using N-grams.

Table 3: Summary of Performance Metrics after Using N-grams & Advanced pre-processing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Naive Bayes with Bag-of-Words** | 0.8420 | 0.8433 | 0.8420 | 0.8418 |
| **Naive Bayes with TF-IDF** | 0.8550 | 0.8566 | 0.8550 | 0.8548 |
| **Logistic Regression with Bag-of-Words** | 0.8560 | 0.8562 | 0.8560 | 0.8560 |
| **Logistic Regression with TF-IDF** | 0.8755 | 0.8761 | 0.8755 | 0.8754 |

Figure 2: Strategy implementation in Line Chart

According to the above tables (Table 2 and Table 3), Logistic Regression with TF-IDF is the best model for conducting sentiment analysis. Because after some extra strategy implementation, it has reached the highest accuracy level compared with others. but Applying N-grams did not have a significant impact on the accuracy of the models.

In the advanced data cleaning methods, I have included a lot of processes. Such as removing special characters and punctuation, removing stop words (frequently in the English language but typically do not contribute much to the sentiment or meaning of a sentence), lemmatizing (process of reducing words to their base or dictionary form) the words, and applying stemming (process of reducing words to their root or stem form) to each word. It could give extra accuracy to the results.

Further analysis and comparison of performance metrics, along with considerations of computational efficiency, interpretability, and generalizability, are recommended to make informed decisions about the most suitable preprocessing techniques and models for sentiment analysis tasks.