Report

Name:- Deepika Hemant Tendulkar

**1. Introduction**

The goal of this project is to classify movie reviews as positive or negative using the K-Nearest Neighbors (KNN) algorithm. This report outlines the implementation details, parameter choices, feature selection, cross-validation process, and the efficiency of the algorithm in terms of runtime. The dataset consists of movie reviews labeled as either positive or negative, and the task is to build a model that can accurately classify unseen reviews.

**2. Instructions on Running the Program**

To run the program, follow these steps:

Ensure you have the necessary libraries installed: numpy, scikit-learn, BeautifulSoup, nltk.

Mount Google Drive if running on Google Colab.

Ensure the training and testing data files are located in the specified paths.

Execute the provided code in a Python environment, ensuring the file paths are correctly set.

The program will output the predicted sentiments for the test data into the specified output file.

**3. Approach**

Preprocessing:-

Text preprocessing was performed using BeautifulSoup for HTML parsing, regular expressions for removing non-alphabetic characters, and NLTK's stopwords for removing common English words.

TF-IDF vectorization was applied to convert text data into numerical features while considering unigrams and bigrams.

Parameter Choices

For TF-IDF vectorization, the following parameters were chosen:

norm='l2': Each output row will have unit norm.

min\_df=0: Ignore terms that have a document frequency lower than the given threshold.

use\_idf=True: Enable inverse-document-frequency reweighting.

smooth\_idf=False: Smooth idf weights by adding one to document frequencies.

sublinear\_tf=True: Apply sublinear tf scaling, i.e., replace tf with 1 + log(tf).

ngram\_range=(1, 2): Consider unigrams and bigrams.

max\_features=9000: Limit the number of features to the top 9000 most frequent.

Cross-Validation:-

Implemented 5-fold cross-validation to evaluate the model's performance.

The average accuracy across all folds was used as the evaluation metric.

KNN Algorithm:-

KNN was implemented to classify test instances based on their similarity to training instances.

Cosine similarity was used as the distance metric.

The number of neighbors (K) was set to 500. The accuracy dropped after 500.

**4. Results**

The results of the experiments conducted during parameter tuning and cross-validation are presented below:

|  |  |  |
| --- | --- | --- |
| **Experiment** | **Parameters** | **Accuracy** |
| Experiment 1 | Default TF-IDF parameters | 0.84 |
| Experiment 2 | Optimized TF-IDF parameters | 0.85 |

**5. Efficiency**

Efficiency of the algorithm was evaluated in terms of runtime. Here are the observations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Experiment | Runtime (Before) | Accuracy (Before) | Runtime (After) | Accuracy (After) |
| Experiment 1 | 45s | 0.84 | - | - |
| Experiment 2 | - | - | 47s | 0.85 |

**6. Conclusion**

In conclusion, the KNN algorithm coupled with TF-IDF vectorization proved to be effective in classifying movie reviews with reasonable accuracy. By optimizing the TF-IDF parameters, a slight improvement in accuracy was achieved. However, there is room for further experimentation with different feature representations and distance metrics to potentially enhance the model's performance. Additionally, considering runtime efficiency, future work could explore dimensionality reduction techniques to handle high-dimensional feature spaces more effectively. Overall, this project provides a solid foundation for sentiment analysis tasks using machine learning algorithms.