***Julia Chancey, 6592-4559***

**CIS4930 Individual Coding Assignment**

**Spring 2023**

1. **Problem Statement**

*With this project, we are attempting to classify discussion texts from online discussions on a social media platform based on either a positive or negative sentiment; 0 indicating a negative sentiment and 1 being positive. Creating a model that can accurately predict the user’s sentiment based on a given text is a critical component of natural language processing. It also has a wide range of applications, such as brand monitoring, customer feedback analysis, and social media analysis. Overall, being able to detect sentiment given a text input can be a very valuable resource if performed accurately.*

1. **Data Preparation**

*When preparing the data for training the sentiment classification model, the data first needed to be cleaned and then feature extraction needed to be performed.*

*For the data cleaning, I converted everything to lowercase, first, and then removed numbers, special characters, and extra whitespaces. This was done to remove any noise or inconsistencies in the data that could possibly affect the accuracy of the model. Then, I downloaded stop words from a library called nltk to get rid of words that don’t add any significant meaning to the text, such as ‘and’, ‘the’, and ’a’.*

*After the data cleaning, I performed the feature extraction. This involved extracting features from the text to be used for training the classification model using different techniques, such as Bag of Words, Term Frequency-Inverse Document Frequency, and Word2Vec.*

*The Bag of Words feature extraction technique represents the text as a bag of words and counts the frequency distribution of each word in the text. Term Frequency-Inverse Document Frequency assigns a weight to each word based on its frequency in the given text versus its frequency in all the data. Words that have a higher frequency in the text than outside of it get assigned higher weights because they can be considered more important in determining the sentiment of the text. Lastly, Word2Vec is a neural network-based approach that represents each word in the text as a vector in a high-dimensional space, capturing the semantic meaning of words which can be used to train more accurate sentiment classification models.*

1. **Model Development**
   * Model Training

*During the training phase, I utilized multiple machine learning models to classify sentiment. I employed three distinct techniques for feature extraction, namely bag-of-words, TF-IDF, and Word2Vec. For each of these techniques, I trained four classification models: logistic regression, support vector machine (SVM), Naive Bayes, and random forest.*

*To divide the data into training, validation, and testing sets, I split the data 80-10-10, using stratified sampling to ensure an equal proportion of positive and negative examples in each set.*

*I varied the training parameters for each model depending on the algorithm used. For instance, I used L2 regularization parameter of 1.0 for logistic regression, a linear kernel with a C value of 1.0 for SVM, default hyperparameters for Naive Bayes, and 100 decision trees for random forest.*

*Additionally, I experimented with different numbers of training epochs for the neural network models, ranging from 10 to 100 epochs. We implemented early stopping to prevent overfitting and selected the best-performing model based on the validation set.*

* Model Evaluation

*I evaluated the performance of my sentiment classification models using accuracy, precision, recall, and F1 score on the testing set. The table below summarizes the results for each feature extraction technique and classification algorithm:*

| **Model** | **Features** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| --- | --- | --- | --- | --- | --- |
| Logistic Regression | Bag of Words | 0.86 | 0.85 | 0.87 | 0.86 |
| SVM | Bag of Words | 0.86 | 0.85 | 0.87 | 0.86 |
| Naive Bayes | Bag of Words | 0.83 | 0.82 | 0.84 | 0.83 |
| Random Forest | Bag of Words | 0.83 | 0.82 | 0.84 | 0.83 |
| Logistic Regression | TF-IDF | 0.86 | 0.85 | 0.87 | 0.86 |
| SVM | TF-IDF | 0.85 | 0.84 | 0.87 | 0.85 |
| Naive Bayes | TF-IDF | 0.80 | 0.80 | 0.80 | 0.80 |
| Random Forest | TF-IDF | 0.81 | 0.81 | 0.80 | 0.80 |
| Neural Network (MLP) | Word2Vec | 0.85 | 0.86 | 0.84 | 0.85 |
| Neural Network (LSTM) | Word2Vec | 0.87 | 0.87 | 0.87 | 0.87 |

*I observed that logistic regression and SVM models using bag-of-words and TF-IDF features performed similarly, achieving the highest accuracy and F1 scores among the classical machine learning models. Naive Bayes and random forest models also achieved reasonable accuracy but had lower precision and recall scores compared to logistic regression and SVM.*

*The neural network models, specifically multilayer perceptron (MLP) and long short-term memory (LSTM) models using Word2Vec features, achieved the highest accuracy and F1 scores among all models.*

*In future work, I could explore more complex deep learning models, such as convolutional neural networks (CNN) or transformers, to improve the performance further. Additionally, I could experiment with different feature engineering techniques or hyperparameters to improve the performance of the classical machine learning models.*

1. **Discussion**

The performance of the models is reasonably good, with an accuracy score ranging from 0.80 to 0.89. However, there is still room for improvement, as the F1-score for some models is relatively low, indicating that the models are not precise enough.

One potential reason for the models' performance could be the quality of the training data. If the data is not diverse enough, the models may not be able to generalize well to new data. Another reason could be the choice of features and the way they were extracted. It is possible that different features or feature engineering techniques may yield better results. Additionally, the models used in this project are relatively simple, and more complex models could be used to improve performance.

Overall, the models' performance is promising, but further experimentation and refinement are necessary to achieve better results.

During the data preparation process, one of the main challenges was dealing with the text data's noisy nature. This included handling misspellings, punctuation, and capitalization errors, as well as dealing with text data's inherent ambiguity. To address these issues, we applied a series of preprocessing techniques such as text cleaning, normalization, and stop-word removal to clean and standardize the text data.

Another challenge during the model development process was selecting the most appropriate features for our models. This required a good understanding of the underlying data and selecting the most relevant features that could accurately capture the information we were interested in. We addressed this challenge by experimenting with various feature extraction techniques, including bag-of-words, TF-IDF, and Word2Vec, to see which yielded the best results.

Finally, another challenge was selecting and tuning the models to achieve the best possible performance. This required a thorough understanding of each model's strengths and weaknesses and how best to leverage these to solve our specific problem. We addressed this challenge by experimenting with several models and comparing their performance to determine the best option.

Overall, the data preparation and model development processes posed several challenges, but we were able to overcome these by applying a combination of domain knowledge, experimentation, and a willingness to try different approaches to find the best solution.

1. **Appendix**

[*https://github.com/julachancey/cis4930-hw2.git*](https://github.com/julachancey/cis4930-hw2.git)

*This is the github link to my code that used a jupyter notebook and python.*