[Applied Analytics Project Week 4 Assignment]

Major: Applied Analytics

Name: Heng Bai, Yuang Guo

Introduction:

This report outlines the process and results of our weekly assignment, which focuses on applying three different text embedding methods—Countvectorizer, TF-IDF, and Word2Vec—to a preprocessed dataset. The main objective is to explore how these embeddings transform raw text into numerical representations that can be used in machine learning models. We will also evaluate the performance of models such as Logistic Regression and Support Vector Machines (SVM) trained on these embeddings. The goal is to compare the effectiveness of each method in capturing meaningful patterns from the text, as well as to analyze the impact of these embeddings on classification accuracy and other performance metrics.

Countvectorizer:

In this assignment, Countvectorizer is applied to the preprocessed text data to transform the textual content into numerical representations. The Count Vectorizer creates a sparse matrix, where each row corresponds to a document (in this case, each row of the dataset), and each column represents a unique word in the corpus. The values in the matrix indicate the frequency of each word in each document. This method is useful in tasks such as text classification, where the presence or absence of certain keywords can help differentiate between categories or labels in the dataset. However, in our dataset, which likely includes a wide range of words with varying frequencies, common words such as “the,” “and,” or domain-specific terms may dominate the representation. This could result in a sparse matrix where many words appear only in a few documents, limiting its effectiveness in distinguishing between different classes. While this technique provides a basic feature extraction approach, it doesn't capture relationships between words or their context within sentences.

TF-IDF (Term Frequency-Inverse Document Frequency):

For our dataset, TF-IDF offers a more refined approach by not only counting word frequencies but also adjusting those counts based on the importance of each word across the entire dataset. Since many words are likely to appear frequently across multiple documents (e.g., common stop words or domain-specific terminology), TF-IDF helps reduce their influence by assigning lower weights to commonly occurring terms and giving more weight to words that are less frequent but more unique to specific documents. In the context of text classification, this is beneficial because it helps the machine learning models focus on words that better differentiate between document categories. For instance, terms that are more relevant to certain classes in the dataset (e.g., specific topics or labels) will be emphasized more than overly common words. This helps improve the performance of classifiers like Logistic Regression and SVM, as it allows the models to better identify the most informative features.

Word2Vec:

In this assignment, Word2Vec provides an advanced approach to capturing semantic relationships between words by generating dense word embeddings based on their context. Unlike Countvectorizer and TF-IDF, which rely solely on word frequency, Word2Vec learns how words are used in context by training a neural network. In the context of our dataset, Word2Vec can help capture more complex linguistic patterns that can be crucial for classification tasks. For example, similar words (or words that tend to appear in similar contexts) will have embeddings that are close to each other in the vector space, which can improve the model's ability to classify documents based on subtle word usage differences. In cases where the dataset contains context-dependent information, such as sentiment or nuanced language, Word2Vec embeddings allow models like Logistic Regression or SVM to make more informed predictions. The primary benefit of Word2Vec for this dataset is its ability to retain the semantic relationships between words, which can lead to improved model accuracy and better generalization across unseen data.

Processing Pipeline:

The code workflow begins with loading the preprocessed dataset from CSV files into pandas DataFrames for both training and testing. Once the data is loaded, three different embedding techniques are applied: Countvectorizer, TF-IDF, and Word2Vec. For Countvectorizer and TF-IDF, the text is converted into a sparse matrix of features where each word or token is assigned a numerical representation based on frequency (for Countvectorizer) or relevance (for TF-IDF). For Word2Vec, dense word embeddings are generated by training a neural network to capture the contextual relationships between words. After applying these embeddings, the transformed data is used to train machine learning models like Logistic Regression and Support Vector Machines (SVM). These models are trained by fitting on the training data (with the respective embedding transformations) and making predictions on the test data, where various metrics are recorded for evaluation.

Results and Observations:

The results across the three embedding methods—Countvectorizer, TF-IDF, and Word2Vec—highlight some notable differences in performance. Typically, models using TF-IDF perform better than those using Countvectorizer due to TF-IDF’s ability to downplay common words and emphasize unique terms, leading to more meaningful feature representations. Word2Vec often achieves the best results, as its word

embeddings capture the semantic relationships between words, allowing the models to better understand context and meaning within the text. In terms of metrics such as accuracy, precision, recall, and F1-score, models using Word2Vec tend to outperform the others, especially in tasks that require nuanced understanding of language.