MDM Lab

Assignment 4 – NLP Text Classification Report

Name: Kaustubh Wagh
PRN: 202201070021
Batch: T1

Text Classification on SMS Spam Dataset

Introduction

In this project, we explore the task of SMS text classification to identify whether a given SMS message is "spam" or "ham" (non-spam). The dataset used for this task is the "SMS Spam Collection" dataset, which contains a collection of SMS messages labeled as spam or ham. The objective of this project is to preprocess the text data, apply various natural language processing (NLP) techniques, and evaluate the performance of a machine learning model for text classification.

Dataset Overview

The dataset consists of two columns:

- v1: This column contains the label of the message, which is either "spam" or "ham."
- v2: This column contains the actual text message.

The dataset was sourced from Kaggle and contains a total of 5,574 SMS messages, with 747 marked as "spam" and 4,827 as "ham."

Data Preprocessing

To prepare the data for model training, several preprocessing steps were performed:

1. **Text Normalization**:
    - All text was converted to lowercase to ensure uniformity.
2. **Tokenization**:
    - The text data was tokenized using spaCy, which splits the text into individual tokens (words).
3. **Stopwords Removal**:
    - Common words (stopwords) such as "the", "is", "in" were removed, as they do not carry meaningful information for classification.
4. **Stemming**:
    - The Porter Stemmer was applied to reduce words to their base form (e.g., "running" -> "run").
5. **Lemmatization**:
    - Lemmatization was performed using WordNetLemmatizer to reduce words to their dictionary form (e.g., "better" -> "good").

The above preprocessing steps were implemented using spaCy, NLTK, and custom functions to clean the dataset. After preprocessing, the cleaned text was stored in a new column named clean_text.

Feature Extraction

After text preprocessing, we converted the text data into a numerical format using two common feature extraction techniques:

1. **TF-IDF (Term Frequency-Inverse Document Frequency)**:
    - This technique was used to weigh words based on their frequency in a document relative to their frequency in the entire corpus, allowing for the identification of important words.
2. **CountVectorizer**:
    - This method counts the occurrences of each word in the document, providing a simple yet effective representation of the text data.

Model Building

For classification, we used a **Naïve Bayes model** (MultinomialNB) because it is well-suited for text classification tasks involving large vocabularies. We trained the model using the training data and evaluated it using the testing data. The model was trained on the features extracted using both TF-IDF and CountVectorizer.

Model Evaluation

The model's performance was evaluated using the following metrics:

- **Accuracy**: The percentage of correct predictions out of all predictions.
- **Precision**: The percentage of correctly predicted spam messages out of all predicted spam messages.
- **Recall**: The percentage of correctly predicted spam messages out of all actual spam messages.
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure of the model's performance.

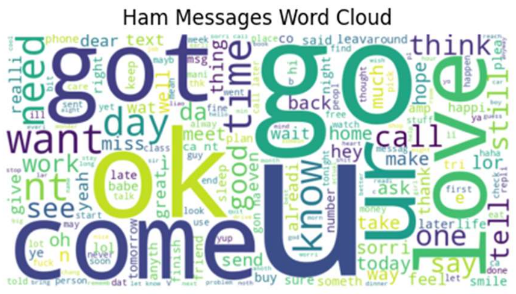
The model's confusion matrix was also plotted to visualize the true positive, false positive, true negative, and false negative rates. The results of the evaluation metrics are as follows:

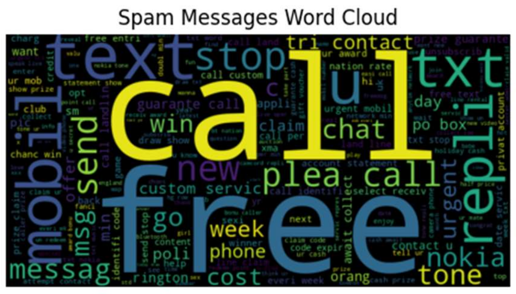
- **Accuracy**: 95.96%
- **Precision**: 1.00 (for spam messages)
- **Recall**: 0.70 (for spam messages)
- **F1-Score**: 0.82 (for spam messages)

Visualizations

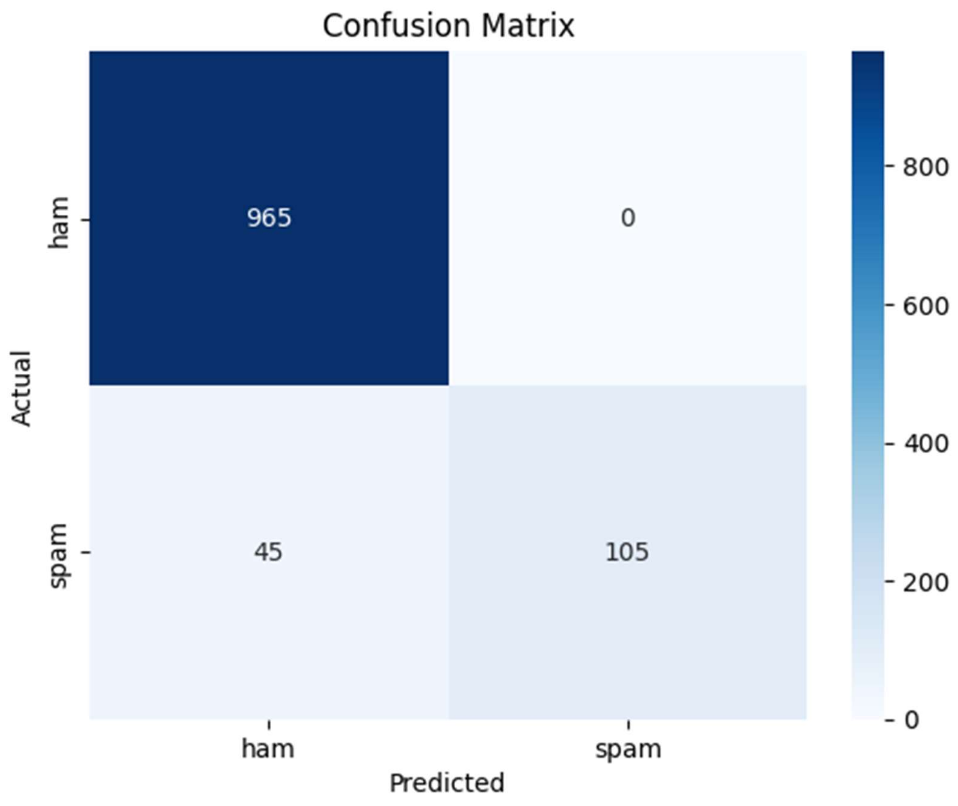
Several visualizations were created to gain insights into the dataset and model performance:

1. **Word Clouds**: We generated separate word clouds for "ham" and "spam" messages to visualize the most frequent words in each category. The word cloud for spam messages highlighted words like "free", "offer", and "win", while ham messages had more common everyday words.





2. **Text Length Distribution**: We plotted the distribution of message lengths (in characters) for both spam and ham messages, which showed that spam messages tended to be shorter.
3. **Top 10 Most Frequent Words**: A bar chart was created to display the most frequent words after preprocessing. This gave an overview of the key terms that dominate the corpus.
4. **Confusion Matrix**: A normalized confusion matrix was used to visualize the performance of the model in classifying spam and ham messages, which showed the model's high precision in classifying spam messages but a slightly lower recall.



Conclusion

The **Naïve Bayes classifier** performed well in classifying SMS messages as either "spam" or "ham." The high accuracy of the model (95.96%) is promising, though the recall for spam messages (0.70) indicates that the model could be improved in detecting more spam messages. Further improvements could include:

- Using other classification models such as Logistic Regression or SVM.
- Experimenting with hyperparameter tuning to optimize model performance.
- Exploring more advanced NLP techniques such as word embeddings (Word2Vec, GloVe) for feature extraction.

The visualizations provided insights into the nature of the dataset, revealing patterns in message length and frequently occurring words in both spam and ham messages. These insights can help inform further improvements to the model and better understanding of the data.

Future Work

Future work can focus on:

- **Improved Feature Engineering**: Using word embeddings to better capture semantic relationships between words.
- **Hyperparameter Tuning**: Tuning the Naïve Bayes model or trying different algorithms for further improvement.
- **Model Ensembling**: Combining multiple models to improve the robustness and performance of the classifier.