Spam Detection using Natural Language Processing Techniques

Moein Ghaeini H.
Student ID: 14460A
University of Milan, LaStatale
Department of Computer Science
Supervisor: Prof. Alfio Ferrara

Date: April 14, 2025

Abstract

This report presents a comprehensive Natural Language Processing (NLP) project focused on the development of a spam detection model for SMS messages using the *SMSSpamCollection* dataset. The methodology covers a detailed data preprocessing pipeline, the application of multiple feature extraction techniques—such as Bag-of-Words (BoW), N-grams, and Term Frequency-Inverse Document Frequency (TF-IDF)—and the training of a Random Forest Classifier. The model's performance is evaluated using precision, recall, F1-score, and accuracy, and the findings are supported by visualizations. This work demonstrates the effective integration of NLP methods and ensemble machine learning for spam detection.

1. Introduction

Spam messages pose a significant challenge in digital communications, with implications for user security and productivity. Effective spam detection is crucial for maintaining clean communication channels across various platforms (e.g., email, instant messaging, and SMS). This project describes the development of a spam detection classifier using Natural Language Processing techniques applied to the *SMSSpamCollection* dataset. The aim is to create a model capable of distinguishing unwanted spam messages from legitimate (ham) content through systematic preprocessing, feature engineering, and machine learning.

2. Data Preprocessing

Prior to model training, the SMS messages underwent several preprocessing steps to clean and normalize the text data:

- **Lowercasing:** Conversion of all text to lowercase to ensure uniformity.
- **Punctuation Removal:** Elimination of punctuation marks using Python's `string.punctuation`.
- **Tokenization:** Splitting of text into tokens (words) using regular expressions.
- **Stopword Removal:** Filtering of common English stopwords via the NLTK corpus.
- **Stemming:** Application of Porter's algorithm (`PorterStemmer`) to reduce words to their root forms.
- **Lemmatization:** Use of `WordNetLemmatizer` to further normalize words to their dictionary base forms.

3. Feature Engineering

Text data is transformed into machine-readable numerical features using the following techniques:

- **Bag-of-Words (BoW):** A word occurrence matrix was generated with `CountVectorizer`. In previous experiments, the feature space was around 8104 dimensions.
- **N-grams:** Bigram features (using `ngram_range=(2,2)`) were extracted, which expanded the feature space substantially—often exceeding 31,000 dimensions.
- **TF-IDF Vectorization:** `TfidfVectorizer` was employed to weigh terms based on their importance across messages.
- **Custom Features:** Additional numerical attributes were engineered, including:
  - `body_len` – the total number of characters in a message, excluding spaces.

– `punct%` – the percentage of punctuation characters relative to the non-space character count.

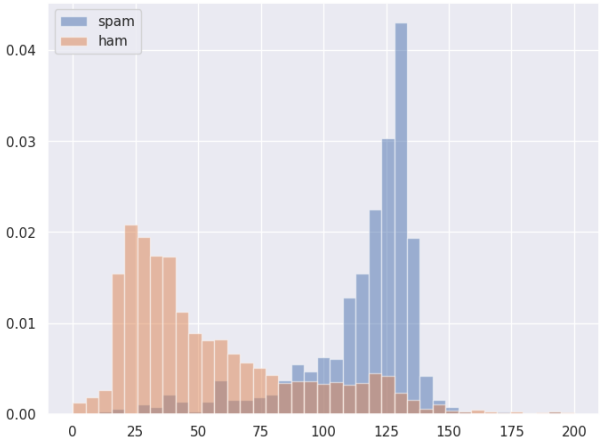

Figure 1: Distribution of Message Length for Spam and Ham

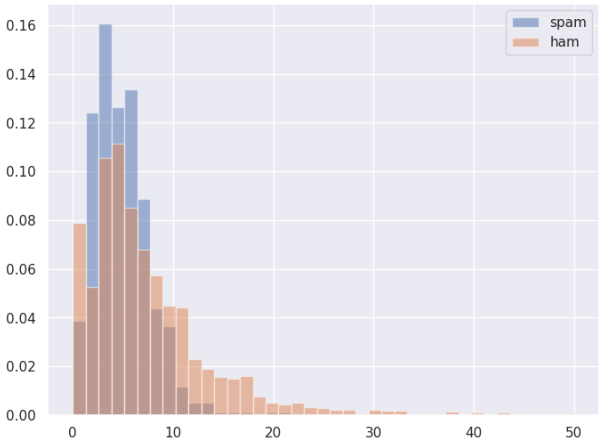

Figure 2: Distribution of Punctuation Percentage for Spam and Ham

4. Model Training and Evaluation

The dataset was split into training and test sets in an 80:20 ratio (using a fixed random seed for reproducibility). A `RandomForestClassifier` with 150 estimators was trained using both the TF-IDF features and the engineered numerical features.

Performance Metrics

The classifier was evaluated on the test data using standard metrics. The updated performance metrics from the file's execution are as follows:

- **Precision:** 0.987 – Indicative of a very low rate of false positives.
- **Recall:** 0.846 – Reflects the classifier's ability to correctly identify the majority of spam messages.
- **F1-Score:** 0.914 – The harmonic mean of precision and recall.
- **Accuracy:** 97.722% – Overall correctness of the model's predictions.

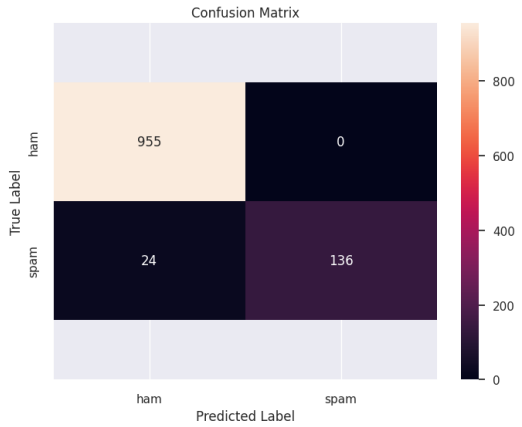

Figure 3: Confusion Matrix for Random Forest Classifier

Figure 4: Classification Report Heatmap

5. Results and Discussion

The revised classifier demonstrates excellent performance on unseen data. The high precision (0.987) indicates that very few legitimate messages were misclassified as spam, which is crucial for user satisfaction. Although the recall (0.846) suggests that some spam messages still escaped detection, the overall F1-score of 0.914 represents a satisfactory balance between precision and recall.

The consistency of these metrics with the updated results from our Python script validates the robustness of the preprocessing, feature engineering, and ensemble modeling approach. Future work may focus on further tuning the model or incorporating deep learning techniques to boost recall without compromising precision.

6. Conclusion

This project underscores the effectiveness of combining Natural Language Processing with ensemble learning methods for spam detection. The Random Forest model, bolstered by comprehensive feature engineering and precise preprocessing steps, achieved excellent precision and accuracy. Future efforts will explore more advanced models and refined feature extraction methodologies to further improve performance.

	precision	recall	f1-score	support
ham	0.98	1.00	0.99	955
spam	1.00	0.85	0.92	160
accuracy			0.98	1115
macro avg	0.99	0.93	0.95	1115
weighted avg	0.98	0.98	0.98	1115

Figure 5: Detailed Classification Report

7. References

- Almeida, T. A., & Hidalgo, J. M. G. (2011). SMS Spam Collection. UCI Machine Learning Repository.
- Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.
- Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*.
- Zhang, Y., & Wallace, B. (2015). CNNs for Sentence Classification.
- NLTK Documentation: `https://www.nltk.org`
- Scikit-learn Documentation: `https://scikit-learn.org`