Size Does Matter: Preprocessing Methods for Enhancing Machine Learning Algorithms in SMS Spam Detection

Elisa Forcada Rodríguez
University of the Basque Country
eforcada001@ikasle.ehu.eus

María García-Abadillo Velasco
University of the Basque Country
mgarciaabadill001@ikasle.ehu.eus

Abstract

This paper investigates preprocessing methods to optimise machine learning algorithms for SMS spam detection, examining 24 scenarios and evaluating five algorithms. The Multinomial Naive Bayes algorithm with more complex preprocessing shows superior precision. However, the overall results suggest that simpler preprocessing generally outperforms complex preprocessing, which is consistent with previous literature and could be explained by the inherently short nature of SMS messages.

1 Introduction

The use of Short Message Service (SMS) has decreased among phone users in the last decade due to the growing popularity of cloud-based messaging platforms. However, SMS still serves as a significant method for identity verification in e-commerce or finances. Now, spam messages have become not only irritating and time-consuming, but also a potential source for phishing attacks and fraud. In this context, automatic spam detention is becoming increasingly important. Therefore, this paper strives to clarify the best techniques for enhancing the performance of various Machine Learning models by focusing on preprocessing methods. For this purpose, the initial experiments conducted by the authors of the corpus will be **partially replicated and augmented**, given the fact that they "did not perform language-specific preprocessing techniques such as stop word removal or word stemming" in accordance with previous literature findings (Almeida et al., 2011).

2 Dataset

The SMS Spam Collection (Almeida et al., 2011) is a public SMS dataset that, to our knowledge, remains the largest publicly available spam detection dataset[1]. The corpus consists of a total of 5,574 SMS messages, of which 86.6% are ham and 13.4% are spam.

2.1 Data acquistion and preparation

The dataset was obtained from the UCI Machine Learning Repository [2] as a text file containing one instance per line. Each line consists of a class tag (ham, spam) separated by tabulation from the raw SMS. This text file was converted into an arff file to enable its use in WEKA. The modifications made to the new arff file included: a) reformatting the file to an arff standard, b) replacing all emojis with the token [EMOJI] to prevent information loss in the tokenization process, c) removing single quotes from the raw messages as they hindered compatibility in WEKA. These new arff files and further work can be found on GitHub.

2.2 Data splitting

After ensuring the dataset compatibility in WEKA, it was splitted into train, development, and test sets according to the following procedure: first, the whole dataset was resampled by randomly setting aside 20% of the data for the test set and 80% for the training set. Following, the same method was applied to the entire training set, resulting in a final allocation of 20% for the development set and 80% for the ultimate training set. Overall, the training set comprises 64%, the development set 16%, and the test set 20%. Duplicate instances were avoided by disabling replacement. Uniformity of classes was not prioritised as the objective of this research is to enhance the models' performance by concentrating on preprocessing techniques, although the

[1](Salman et al., 2022) announced the publication of a dataset consisting of 153,551 SMSes for research purposes, which is still not available online

[2]https://archive.ics.uci.edu/dataset/228/sms+spam+collection

matter of bias remains a interesting aspect for future research.

	Partition			
	Train	**Dev**	**Test**	**Total**
Instances	3565	893	1,116	5,574
Tokens	66,876	16,449	20,485	103,810
nom class ham	3,088	773	966	4,827
nom class spam	477	120	150	747

Table 1: Tokenization generated by the NLTK library. The remaining attributes are 'String' class. Further information on Github.

The quantitative analysis performed using Python is displayed in Table 1, conducted during the Exploratory Data Analysis. This exact tokenization was not employed in any subsequent experiment. Additional information on string vectorization is provided in section 3.2.String representation.

2.3 Exploratory Data Analysis

A Exploratory Data Analysis (EDA) was performed on the train and development datasets in order to get a better understanding of the training data. In Figure 1 and Figure 2, a visual summary provides a clear overview of the central tendency, spread, and presence of outliers in the distribution of SMSes lengths.

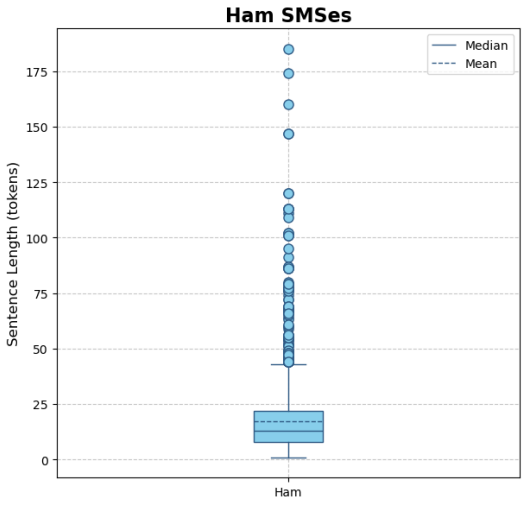

Figure 1: Boxplot illustrating the length of ham instances in percentiles along with their outliers.

We observe a notable distinction in the average length of ham and spam SMS messages, with ham SMSes averaging 17 tokens and spam SMSes averaging 28 tokens. Additionally, our findings indicate that 50% of ham instances have lengths ranging from 8 to 22 tokens, while an equivalent percentage of spam instances falls within the range of 25 to 32

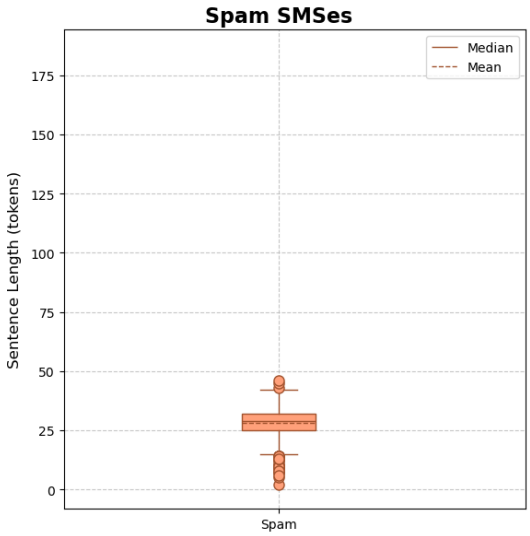

Figure 2: Boxplot illustrating the length of spam instances in percentiles along with their outliers.

tokens. Interestingly, our analysis shows a higher incidence of outliers above the upper quartile for ham SMS compared to spam, with the longest ham SMS being four times longer than the longest spam SMS. However, spam outliers are mostly below the lower quartile. This may explain why previous literature does not recommend sophisticated preprocessing methods for SMS, as its shorter length (compared to e.g. emails) poses a challenge to ML algorithms. Applying these methods reduces the token length and may therefore increase the challenge.

3 Experiments

3.1 String representation

The commonly used method binary bag of words (BOW) was applied in WEKA to signify whether words, considered as attributes, are present (denoted by 1) or absent (denoted by 0) in the various instances of the corpus (Juluru et al., 2021). The features of each instance are stored in a vector, which is used as input for machine learning algorithms. The created vectors are nonsparse (indicating only the present attributes in an instance) in order to to save computational space.

3.2 Preprocessing

Four hiperparameters for preprocessing methods were considered (tokenization, stopwords, stemming, recasing) and applied in WEKA. The explanation of each hiperparameter option is provided below.

- Tokenization 1: tokens are separated by dots, commas and colons.
- Tokenization 2: sequences of characters that are separated by spaces, tabs, returns, colons, periods, commas and hyphens are grouped together.
- Stopwords 1: without stopwords.
- Stopwords 2: Rainbow stopwords list is applied (Kachites, 1996).
- Stopwords 3: this option removes non-alphabetic characters (numbers, special characters...).
- Stemming 1: without stemming.
- Stemming 2: Iterated Lovins Stemmer, an extension of Lovins's algorithm, is applied. This stemmer reduces the word to its stem if it has more than two characters until there is no further change (Lovins, 1968).
- Recasing 1: without recasing.
- Recasing 2: all words are converted to lowercase.

By conducting all possible combinations of these options for each hyperparameter, we obtained 24 preprocessing results. This can be formalised into the following function, where L denotes the different options for each preprocessing technique, taken as hyperparameters, with $H = 4$:

$$\text{Number of cases} = \prod_{i=1}^{H} L_i$$

3.3 Algorithms

The algorithms that produced the most favourable results in the original paper were chosen for our training in WEKA. These were **LibLinear** (closely resembling Linear Support Vector Machine, unfortunately not accessible in WEKA at the time), **Naive Bayes**, and **Multinomial Naive Bayes**. Additionally, it was deemed worthwhile to incorporate **Random Forest** and **Random Tree**, as the authors employed comparable decision tree-based algorithms.

3.4 Results

By training the 24 different preprocessing cases using the 5 selected algorithms in WEKA, 120 results were obtained. To compare these outcomes, precision was considered as one of the most relevant metrics, because it measures the number of spam messages identified as spam (true positives: TP) compared to the total number of messages classified as spam (correct and incorrectly). A higher precision indicates that fewer ham messages will be mistakenly categorised as spam (false positives: FP). A spam message incorrectly classified as ham is a minor problem, as the user is simply required to delete it. On the other hand, a ham classified as spam can be unacceptable, as it implies the loss of potentially important information, particularly when spam is automatically deleted (Guzella and Caminhas, 2009).

Another considered metric was the ROC area, which illustrates the performance of a classification model across all classification thresholds when plotted. The ROC curve depicts two key parameters: the true positive rate and the false positive rate. The area under the ROC curve quantifies the two-dimensional area covering the entire curve, ranging from 0.5 (indicating a random classifier) to 1 (representing the best possible classifier).

The different metric results from all the 120 outcomes is shown in Figure 3, with cold and warm colours representing the different tokenization and stopwords techniques, and figures (filled or empty squares and circles) representing the type of stemming and recasing methods. This combination was selected randomly for the purpose of visualization.

Overall, the Multinomial Naive Bayes approach achieved the most favourable precision results, which we consider one of the most relevant metrics. It is worth noting that there was another combination of hyperparameters that yielded the highest TP Rate, indicating that that model classified every instance as spam and thus performed the worst in FP Rate. Aditionally, cases with limited preprocessing techniques (warmer colors, e.g. solely using lowercase letters or stemming without the application of stopwords or special tokenization) achieved in general better results.

After obtaining all these results, the top five precision algorithms were subjected to the information gain score preprocessing mechanism (not shown in Figure 3). Different thresholds of 0.001, 0.004, and 0.05 were applied. However, the 15 results obtained in all five top cases were worse than those without information gain methods. We did not use this approach in every experiment be-

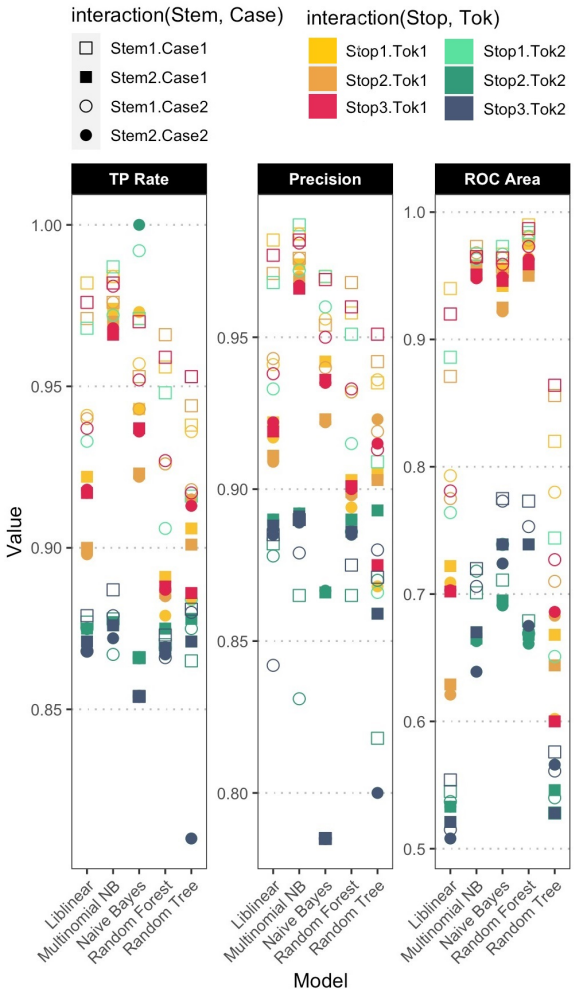

Figure 3: Performance of all 120 outcomes.

cause we considered it would open up a new line of research. Furthermore, these results show that information gain is not necessarily beneficial in settings where instances are short. Interestingly, the token [EMOJI] was always among the 75 most relevant attributes based on information gain.

Finally, the top-performing result (Multinomial Naive Bayes tokenization 2, stopwords 1, stemming 1, and recasing 1) was evaluated with the test set, obtaining a very slight decrease in its performance, achieving an precision of 0.983 and a ROC Area of 0.983.

4 Conclusions

This paper examines 24 preprocessing scenarios to optimize machine learning algorithms for SMS spam detection, evaluating five algorithms. Despite the Multinomial Naive Bayes algorithm demonstrating superior precision with complex preprocessing, the overall results support the idea that simpler methods often outperform complexity. The inherent brevity of SMS poses a challenge for machine learning, and additional preprocessing may contribute to this trend.

Beside this, it should be noted that while some hyperparameter combinations perform admirably in terms of TP rate, they fare poorly in terms of PF: this underscores the importance of considering multiple metrics when evaluating and comparing classifiers. In this case, precision and ROC area were considered relevant metrics because of the relatively low absolute spearman correlation (0,71) among them.

5 Discussion

This paper examines different preprocessing approaches to boost the efficiency of SMSes classification algorithms. Conducting further research on additional preprocessing methods, particularly those associated with information gain, would be advantageous. Investigating other hyperparameters, like possible dataset partitions and different alternatives for string representation, could also unveil new areas for research.

References

Tiago A. Almeida, José María G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering*, DocEng '11, page 259–262, New York, NY, USA. Association for Computing Machinery.

Thiago S. Guzella and Walmir M. Caminhas. 2009. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222.

Krishna Juluru, Hao-Hsin Shih, Krishna Nand Keshava Murthy, and Pierre Elnajjar. 2021. Bag-of-words technique in natural language processing: A primer for radiologists. *RadioGraphics*, 41(5):1420–1426. PMID: 34388050.

Kachites. 1996. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Http://www.cs.cmu.edu/ mccallum/bow.

Lovins. 1968. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31.

Muhammad Salman, Muhammad Ikram, and Mohamed Ali Kaafar. 2022. An empirical analysis of sms scam detection systems. *arXiv preprint arXiv:2210.10451*.