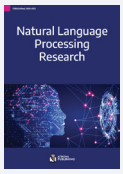
ATHENA PUBLISHING

Natural Language Processing Research
Volume 3, Issue 1–2, December 2023, pp. 1–13
DOI: https://doi.org/10.55060/j.nlpre.230606.001, ISSN (Online): 2666-0512
Journal home: https://www.athena-publishing.com/journals/nlpre



RESEARCH ARTICLE

The Impact of Applying Different Pre-Processing Techniques on Swahili Textual Data Using Doc2Vec

Bernard Masua[1,*], Noel Masasi[1,a], Hellen Maziku[1,b], Betty Mbwilo[1,c]

[1]*College of Information and Communication Technologies (CoICT), University of Dar es Salaam, Dar es Salaam, Tanzania*

[a]*Email: noeliasmasasi@gmail.com*
[b]*Email: maziku.hellen@udsm.ac.tz*
[c]*Email: engbettie@gmail.com*

ABSTRACT

Data pre-processing is an important step in machine learning text classification as it improves data quality and hence improves performance of trained algorithms. We experimentally compare the following pre-processing techniques: punctuation removal, lowercasing, typos replacement, slang replacement and stop-word removal on a Swahili short message service (SMS) dataset for topic classification. Different machine learning algorithms are applied such as Random Forest, Stochastic Gradient Descent, RNN LSTM Unidirectional, RNN LSTM Bidirectional and Support Vector Machine. We analyze the impact of the pre-processing techniques on classification accuracy and f1-score. Our experiments show that all pre-processing steps, when applied separately, have a positive impact on the performance of all evaluated classification algorithms. Among all experimented pre-processing steps, stop-word removal has the highest impact on performance of both accuracy and f1-score metrics. Also, of all evaluated algorithms, Random Forest and Stochastic Gradient Descent are the most positively affected with pre-processing steps.

ARTICLE DATA

Article History
Received 30 March 2023
Revised 15 May 2023
Accepted 22 May 2023

Keywords
Natural Language Processing
Text pre-processing
Swahili language
Stop words
Slang
Typos
Machine Learning

HIGHLIGHTS

- The study aims to evaluate the performance of classification algorithms which are vital in automating error-prone manual work.
- The study illustrates the importance and effects of different pre-processing steps for Swahili textual data.
- This article will enable future researchers to decide which pre-processing steps for Swahili textual data are best for their respective machine learning tasks.

1. INTRODUCTION

Natural Language Processing (NLP) is a field of Artificial Intelligence that gives machines the ability to read, understand and derive meaning from human languages. Swahili is a language spoken by the majority in East and Central Africa. However, Swahili is considered as a low recourse language and there is not enough research on machine learning techniques that utilize textual data based on the Swahili language. Any language's basic sentence structure employs both a subject-verb (SV) and a subject-verb-object (SVO) formula. But they differ when a sentence gets more complex, even for some relatively simple sentences. Language structure differs on word order, use of unique words, time and environment. Swahili is distinguished from other languages by its basic syllable structure, which includes no consonant clusters, no final consonants and the addition of a vowel to loanwords that finish in a consonant [1]. This article uses a Swahili dataset collected from SMS platforms on different topics ranging from education, nutrition, health and corona, among others. As the number of collected messages increases there is a need to automatically classify each incoming message into its corresponding topic category.

SMS platforms are used to collect opinions that can be used to generate insights on different matters. As people express their views and opinions on certain things, they

*Corresponding author. Email: bhrmasua@gmail.com

usually send SMS messages that contain significant amounts of noise. We define noise as data that do not contain any useful information for the analysis at hand and that mislead machine learning algorithms into incorrect decisions. Noise can be punctuations, stop words, spelling and typos, slang and non-alphabetic characters which affect data quality. Data quality is a feature enhanced through a series of data pre-processing steps. Data pre-processing is critical in preparing Swahili textual data for NLP classification tasks. It can improve the accuracy and efficiency of the classification model, while also making it easier to analyze the textual data and extract meaningful insights. Hence, the following pre-processing steps should be considered: removing punctuations and other non-alphabetic characters, converting all text into lower case, replacing common typos with proper words, replacing common Swahili slang with proper words, and removing stop words. These steps were applied separately and then each algorithm was re-trained and re-evaluated to assess the effects.

The purpose of this study is to assess common pre-processing techniques from previous studies and to evaluate their impact on the performance of classification algorithms. Accuracy and F1 score metrics are used to evaluate the performance of algorithms. Document to Vector (Doc2Vec) [2] is used as a feature extraction approach for all algorithms implemented in this study. Fine-tuned models of each algorithm are used as a base to compare the impact of each pre-processing step. The contributions of this article are as follows:

1. The study contributes to the body of knowledge that, among all evaluated algorithms, Random Forest (RF) and Stochastic Gradient Descent (SGD) are more affected by pre-processing steps on Swahili textual data, whereas LSTM Unidirectional and Support Vector Machine (SVM) are less affected by these pre-processing steps.
2. All pre-processing steps tested show a positive impact on the performance of the evaluated classifiers, but a higher impact was observed when stop words were removed from the Swahili textual data. Hence this should always be considered during pre-processing.
3. Research developed and contributed common Swahili Stop Words dataset containing 254 unique words [3], common Swahili Slangs dataset containing 234 words for slang and their respective Swahili proper words [4] and common Swahili Typos dataset containing 431 misspelled words their respective Swahili proper words [5].

The organization of this paper is as follows. Section 2 reviews some of the related literature. In Section 3, experiments are discussed in terms of the dataset used, the pre-processing methods, the feature selection approach, the classification algorithms and the evaluation metrics deployed. Section 4 discusses the findings and the results. The conclusions are detailed in Section 5, followed by compliance statements and the list of references.

2. RELATED WORKS

Etaiwi et al. [6] evaluated the impact of applying different pre-processing steps on review spam detection. Among the steps used were Part-of-Speech (POS) tagging, n-gram term frequencies, stemming, and stop word and punctuation marks filtering. The study used a dataset proposed in [7,8] consisting of about 1600 reviews. The authors compared truthful and deceptive positive and negative reviews for hotels found on the TripAdvisor website. The authors observed that these steps affect the overall accuracy of the review spam detection task which was carried out by training and evaluating the performance of the Naïve Bayes (NB), Support Vector Machines (SVM), Decision Tree, Random Forest (RF) and Gradient Boosted Trees algorithms. Results showed that each different pre-processing step may have a positive or a negative impact on the performance of each algorithm. However, the authors did not consider replacing common slang and typos with proper words, although these two steps depend on the content and source of the dataset used.

A paper written by Işik et al. [9] about the impact of text pre-processing on the prediction of review ratings by using the K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), Stochastic Gradients Descent (SGD), Naïve Bayes Classifier (NB) and Support Vector Machine (SVM) classifiers of the Natural Language Toolkit (NLTK) to get accuracy results for all types of pre-processing methods. The study used a real e-commerce dataset extracted from Yelp in June 2018 consisting of 10,000 restaurant reviews aiming to analyze the effects of pre-processing methods when finding the star ratings of restaurants by analyzing the reviews. The study highlights that text pre-processing has a remarkable impact on the performance of classifiers. Some pre-processing methods have a positive or a negative effect on the classification accuracy, while others have a neutral effect. Also, the order of applying the pre-processing methods matters. The authors did not consider replacing common slang with proper words as one of the text pre-processing steps.

In [10] the impact of pre-processing steps on the accuracy of machine learning algorithms in sentiment analysis was evaluated by using a dataset extracted from the Twitter API with the KNIME tool that filtered only tweets written in the English language. For the sentiment analysis task, SVM, NB and Maximum Entropy (MaxE) algorithms were used. The results illustrated that the accuracy of models generated by the SVM and NB algorithms was improved positively, while that of MaxE remained constant after

applying pre-processing steps. This article calculated the accuracy of the three machine learning algorithms before and after applying the pre-processing steps without considering each step separately to evaluate its effect on the trained algorithm performance.

3. MATERIALS, THEORY AND METHODS

This section contains a brief discussion on the dataset used for the evaluation, the pre-processing steps, the feature selection approach, the classification algorithms and the evaluation measures employed within this study.

3.1. Dataset

The Swahili textual data used in this study consist of short text messages received on U-Report Tanzania. U-Report has over 22 million subscribers worldwide and over 170,000 subscribers in Tanzania, and is a global platform that allows young people to express their views on topics across various fields such as Health, Education, Menstrual Hygiene, Corona, WASH, Nutrition, HIV and Violence against Children. The dataset contains 64,390 rows and 3 columns which are Topic Number, Topic Name and Text.

Datasets were split in a ratio of 75% by 25% for training and testing respectively. To ensure that the training dataset included all possible patterns used for defining the problem and extended to the edge of the modeling domain, we first generated data frames for each class of text, then split each class using the Sklearn function by a 75:25 ratio with enabled random state and combined the resulting outputs to form the training dataset and testing dataset. Table 1 shows the number of sentences or rows each topic contributes to the labeled dataset. Fig. 1 shows the percentage contribution of each topic in rows of the labeled dataset.

Topic Name	Topic	Text
Corona	9	13,832
Education	3	16,682
HIV AIDS	4	4,829
Health	1	4,197
Menstrual Hygiene	7	3,768
Nutrition	2	4,984
Others	8	2,039
U-Report	10	4,758
Violence Against Children (VAC)	5	7,369
WASH	6	1,932

Table 1. Topics labeled statistic.

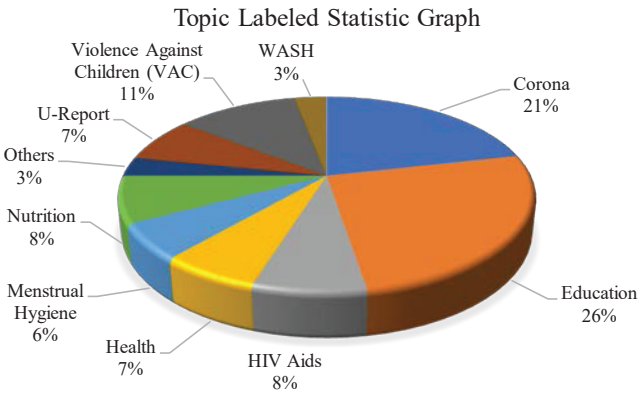


Figure 1. Dataset topic labeled in percentages.

3.2. Pre-Processing Methods

NLP pre-processing steps vary depending on the nature of the data and the intended machine learning task. This study uses a Swahili dataset collected from youth through an SMS platform with the task of categorizing each SMS into an identified topic category. The following pre-processing steps were considered to prepare the dataset for training and testing classification algorithms.

3.2.1. Removing Punctuations and Other Non-Alphabetic Characters

Punctuations are noisy data which may mislead trained algorithms to make wrong decisions during topic classification. Hence punctuations should be removed during data pre-processing [11]. Also, removing punctuations and non-alphabetic characters will provide relief to processors during training and reduce the overall processing time and resources.

3.2.2. Converting Text to Lowercase

Uppercase and lowercase characters or words are treated differently by computers. Hence the same words or characters with different cases will be assigned different vectors during the vectorization step. By doing so, the same words are merged to have clean token information and reduce dimensionality. This is performed at an early stage of pre-processing, to help other pre-processing stages perform well and to reduce the effect of missing the word with letters written in different cases.

3.2.3. Replacing Common Typos

The same words with slightly different spelling are treated differently by computers and are therefore assigned different vectors during the vectorization step. Performing spelling corrections allows the same words to be merged to have clean token information and reduce dimensionality. This study applies a typos dataset from [5] to replace misspelled Swahili words with proper words.

3.2.4. Replacing Common Swahili Slang

Words that are regarded as very informal are known as slang. Slang depends on location, on particular context or on groups of people and is treated differently by computers and thus assigned different vectors during the vectorization step. Performing replacement of slang means that words with similar meaning are merged to have clean token information and reduce dimensionality. This study applies a slang dataset from [4] to replace slang with proper words.

3.2.5. Removing Stop Words

The common Swahili stop words dataset from [3] consists of a list of words which do not add much meaning to a sentence. Hence these can be ignored without sacrificing the meaning of sentences. Stop words are removed because they do not contribute to finding the context or true meaning of a sentence. Removing stop words reduces dimensionality during training and provides relief to processors, hence reducing overall training time.

3.3. Text Representation Techniques

Text features can be extracted by using different methods. This study uses Doc2Vec [2], also known as Para2Vec, which is an NLP technique used to represent documents as a vector and is a generalization of the Word2Vec method. Doc2Vec creates a numeric representation of a document like a sentence or a paragraph, regardless of its length. Documents, unlike words, do not come in logical structures. So instead of using the Word2Vec approach, another vector was added to represent a paragraph or sentence [12].

To generate a predictive model, this technique creates a distributed semantic representation of words in the document that is trained in the context of each word. It learns how to connect documents and words by learning a conceptual representation of a document from a Swahili corpus of documents. In a process of vectorizing Swahili paragraphs/sentences, every paragraph/sentence is mapped to a unique vector. During training, the model learns vectors which are a semantic representation of the documents. To represent each document, the paragraph and word vectors are averaged or concatenated to predict the next word in context [13]. The averaged or concatenated vectors of paragraphs/sentences are used to build the vocabulary from a sequence of sentences. This represents the vocabulary of the model which keeps track of all unique Swahili words.

After training, the model is inspected to ensure that it learned all the words and their contextual meaning. Validation is done by generating most similar words using the most_similar function in the Gensim library. When a word is passed, the model lists all words in the document which contextually resemble the passed word as illustrated in Table 2.

3.4. Classification Algorithms

The scope of this research involves checking the impact of pre-processing techniques on the performance of the following classification algorithms: Random Forest, Stochastic Gradient Descent, RNN LSTM Unidirectional, RNN LSTM Bidirectional and Support Vector Machine.

3.4.1. Random Forest

Random Forest (RF) [14], as its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes the model's prediction. The trees in RF protect each other from their individual errors: while some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction [15].

3.4.2. Stochastic Gradient Descent (SGD)

Gradient Descent [16] is an iterative algorithm that starts from a random point on a function and travels down its

S/No.	Word	Near/Similar Word With Weight
1	ajira	[('mikopo', 0.6587635278701782), ('mitaji', 0.6048551797866821), ('pensheni', 0.5972285270690918), ('motisha', 0.5750974416732788), ('leseni', 0.5745390057563782), ('huduma', 0.5536848902702332), ('nguvukazi', 0.5497235059738159), ('uwakilishi', 0.5494844913482666), ('raslimali', 0.5410809516906738), ('mapato', 0.5391908884048462)]
2	kuajiriwa	[('kuandikishwa', 0.6560262441635132), ('kustaafu', 0.6464136838912964), ('kuwaajiri', 0.6243953704833984), ('kuajiri', 0.6227836608886719), ('kujisajili', 0.6226930618286133), ('kujiendeleza', 0.616544723510742), ('kujiandikisha', 0.6145368814468384), ('kuhitimu', 0.601360023021698), ('kusajiliwa', 0.5967570543289185), ('kujiajiri', 0.585351288318634)]
3	kujiajiri	[('wajiajiri', 0.6244261264801025), ('kujiari', 0.60692298412323), ('kulima', 0.5857206583023071), ('kuajiriwa', 0.585351288318634), ('kujitegemea', 0.5762639045715332), ('mitaji', 0.5622026920318604), ('kujiendeleza', 0.559045135974884), ('kujihudumia', 0.5459901690483093), ('kimtaji', 0.5443302392959595), ('kuviendesha', 0.5207050442695618)]

Table 2. Top-10 similar/related words.

slope in steps until it reaches the lowest point of that function. SGD randomly picks one data point from the whole data set at each iteration to significantly reduce the computations [17].

3.4.3. RNN

Recurrent Neural Network (RNN) [18] is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature since it executes the same function for each data input, and the current input's outcome is dependent on the previous computation. RNN algorithms are included in this study because they are important and powerful in most NLP works. RNNs are designed to use sequential data, and the result is enhanced by storing past calculations. RNNs have a memory function that preserves previously calculated information. RNNs with LSTM units have the advantage of being able to learn long-term dependencies by altering the information in a cell state using three separate gates [19].

Bidirectional and Unidirectional RNN LSTM are the two forms of RNN LSTM used in this study. Bidirectional LSTM will handle generated Swahili text vectors as inputs in two ways: from the past to the future and from the future to the past. Bidirectional LSTM differs from unidirectional in that it uses LSTM that runs backward to preserve information from the future, and by combining the two hidden states, it can maintain information from both the past and the future at any point in time [20].

3.4.4. Support Vector Machine

The objective of the Support Vector Machine algorithm [21] in this research is to find a hyperplane in an N-dimensional space, where N is the number of features from Swahili text vectors that distinctly classifies the data points into mentioned topics. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different topics. Also, the dimension of the hyperplane depends upon the number of features. In this study the number of input features is 10, so the hyperplanes generated to distinguish 10 topics are just 9 lines. To separate classes of data points, there are many possible hyperplanes that could be chosen. The objective is to find a plane that has the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future text data points can be classified with more confidence [22].

3.5. Evaluation Metrics

For performance evaluation, we use accuracy and F-Measure (f-score) since f-score measures recall and precision at the same time. In order to have a testing dataset, the dataset is split into training and testing sets with a ratio of 3:1 respectively. The dataset splitting process was done with the consideration that each topic must be in both sets with the same ratio of 3:1. This will reduce the chance of overfitting and underfitting that may occur when using a random split method. The Swahili testing dataset was generated with 25 percent of about 16,098 sentences from each topic and was used for testing each generated Swahili model. The equation for calculating the f-score is [23]:

$$F-measure = \frac{2 * Recall * Precision}{Recall + Precision}$$

where

$$Recall = \frac{TP}{TP + FN} \quad \text{and} \quad Precision = \frac{TP}{TP + FP}$$

and the equation for calculating accuracy is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

True Positive (TP): the model predicted positive and it is true.
True Negative (TN): the model predicted negative and it is true.
False Positive (FP) (Type 1 Error): the model predicted positive and it is false.
False Negative (FN) (Type 2 Error): the model predicted negative and it is false [23].

3.6. Hyper-Parameters Tuning

Hyper-parameters are those parameters that are not altered during the learning process. Parameter tuning is used to configure a model or algorithm while decreasing the cost function. This is simply an optimization loop built on top of a machine learning model learning to discover the set of hyper-parameters that lead to the lowest error on the validation set in the tuning process. There are three different types of hyper-parameter optimization algorithms: comprehensive space search that includes all available search options, surrogate models that predict the local lowest validation loss when hyper-parameters are fitted on earlier trials, and the third category which combines the two previous ideas [24].

In this study, we employ the first category to fine-tune parameters on algorithms by combining two hyper-parameter optimizer techniques: Random Search and Grid Search. Random Search is a type of hyper-parameter optimization algorithm used in space category exhaustive search that randomly samples the search space rather than discretizing it with a cartesian grid. It has a time budget in the sense that the number of trials to end the search must be defined [25]. Grid Search is a type of hyper-parameter optimization algorithm used in space category exhaustive search, where the complete search space is discretized as the cartesian product of each

hyper-parameter. The algorithm then performs parallel learning for each of the hyper-parameter combinations, evaluates their performance and chooses the best one [26]. Table 3 shows fixed tuned parameters for each algorithm which are later used in the study to observe the impact of Swahili text pre-processing on machine learning classification of topics.

4. RESULTS AND DISCUSSION

In the experimentation phase, Doc2Vec is implemented using Sklearn and is used to extract features for all classification algorithms. The Random Search method is used to determine a set of optimal hyper-parameters. Then, the generated set of parameters is used as input for the Grid Search method to obtain hyper-parameters with the lowest validation errors. This process is done to reduce computational resources and the curse of dimensionality. Table 4 shows improvements in accuracy and f1-score after parameter fine-tuning. Swahili models generated from this study can be accessed through a GitHub repository [27].

4.1. Pre-Processed Dataset Analysis

First each pre-processing step on the dataset was performed separately and later on the combination of steps was done all together. The total number of words in the dataset was 94,153 with 606,009 characters. The punctuation removal step affects 0.63% of all characters present in the dataset which is equal to 3,814 characters. With the typos correction step 650 words were affected which is equal to 0.69%. For slang correction 0.05% of words, which is equal to 47 words, were affected. Stop word removal affected 15.67% of words which is equal to 14,754 words. Also, after performing all the steps, 25,831 words which is equal to 27.44% of all the words, and 92,897 characters which is equal to 15.31% of all the characters, were affected.

4.2. Effect of Fine-Tuning

By using the Doc2Vec text representation technique, experiments show that without hyper-parameter fine-tuning, the model generated by the RNN LSTM Unidirectional algorithm outperforms other algorithms and registers an accuracy of 79.33% and f1-score of 80.26%. After making improvements to the model by fine-tuning, the RNN LSTM Unidirectional model still outperforms other models with an accuracy of 81.50% and f1-score of 82.47%. The improvements by fine-tuning vary from model to model with SVM registering the highest improvement in both accuracy and f1 score by 20.57% and 17.94% respectively, as shown in Table 4. When employing Swahili textual data, the fine-tuning process has the greatest beneficial impact on accuracy and f-score for all models.

4.3. Effect of Removing Punctuations

Table 5 shows the results obtained after removing punctuations which are ! ” # $ % ’ ( ) *+ , – . / : ; < = > ?

S/No.	Algorithm	Parameters
1	SGD	alpha = 0.001, loss = 'hinge', max_iter = 100, random_state = 42, tol = None
2	SVM	C = 1.2, kernel = 'rbf', probability = True, random_state = 0
3	RF	min_samples_split = 5, criterion = 'gini', n_estimators = 10000, n_jobs = –1, random_state = 0
4	RNN LSTM Uni	Max_NB_Words = 100000, Max_Sequence_Length = 950, Embedding_Dimension = 300, LSTM (100, SpatialDropout = 0.2, recurrent_dropout = 0.2), Dense = 10, activation = 'softmax', loss = 'categorical_crossentropy', optimizer = 'adam'
5	RNN LSTM Bi	Max_NB_Words = 100000, Max_Sequence_Length = 950, Embedding_Dimension = 300, Bidirectional (LSTM (100, return_sequences = True, dropout = 0.50), merge_mode = 'concat'), Dense = 100, activation = 'softmax', loss = ' sparse_categorical_crossentropy, optimizer = 'adam'

Table 3. Tuned parameters for classification algorithms.

Algorithm	Accuracy without fine-tuning	F1-score without fine-tuning	Accuracy with fine-tuning	F1-score with fine-tuning	Accuracy improvement after fine-tuning (%)	F1-score improvement after fine-tuning (%)
SGD	0.6672	0.6577	0.7654	0.7733	14.72	17.58
SVM	0.6553	0.6733	0.7901	0.7941	**20.57**	**17.94**
RF	0.5814	0.6336	0.6846	0.7193	17.75	13.53
LSTM Uni	0.7933	0.8026	**0.8150**	**0.8247**	2.74	2.75
LSTM Bi	0.7756	0.7818	0.8070	0.8024	4.05	2.63

Table 4. Improvements made to the model by fine-tuning.

@ [\ ] _‘ { | } and are 0.63% of all characters in the dataset. Results show that punctuations removal has a positive effect on the accuracy and f-score for all the trained models. The improvements vary from model to model with SGD registering the highest improvement in accuracy of 1.33% and LSTM Bi achieving the highest improvement in f1-score of 0.86%.

4.4. Effect of Lowercasing

Table 6 shows the results obtained after changing all characters to lowercase. 14.31% of all words are affected by this step, which leads to a reduction in the number of vectors generated during vectorization since both lowercase and uppercase characters in the same words are mapped to a single vector. Results show that lowercasing has a positive effect on the accuracy and f-score for all the trained models. The improvements vary from model to model with SGD registering the highest improvement in accuracy of 2.48% and RF achieving the highest improvement in f1-score of 1.65%.

4.5. Effect of Replacing Typos With Proper Words

Table 7 shows the results obtained after replacing common Swahili typos with proper Swahili words. According to the dataset used the word “maswali” can be commonly misspelled and written in four different ways which are “maswal”, “maswar”, “maxwal” and “maxwali”. Without replacing typos, all these five words will be represented with five different vectors. So correcting typos leads to a reduction in the number of vectors generated during vectorization and 0.69% of words are affected by this step. Results show that replacing typos with proper words has a positive effect on the accuracy and f-score for all the trained models. The improvements vary from model to model with SGD registering the highest improvement in accuracy and f1-score of 3.23% and 2.66% respectively.

Algorithm	Without Pre-Processing		Punctuations Removal		Accuracy improvement (%)	F1-score improvement (%)
	Accuracy	F1-score	Accuracy	F1-score		
SGD	0.7654	0.7733	0.7756	0.7791	**1.33**	0.75
SVM	0.7901	0.7941	0.7982	0.7994	1.03	0.67
RF	0.6846	0.7193	0.6864	0.7211	0.26	0.25
LSTM Uni	0.8150	0.8247	**0.8161**	**0.8267**	0.13	0.24
LSTM Bi	0.8070	0.8024	0.8089	0.8093	0.24	**0.86**

Table 5. Results after removing punctuations.

Algorithm	Without Pre-Processing		Lowercasing		Accuracy improvement (%)	F1-score improvement (%)
	Accuracy	F1-score	Accuracy	F1-score		
SGD	0.7654	0.7733	0.7844	0.7801	**2.48**	0.88
SVM	0.7901	0.7941	0.7996	0.8012	1.20	0.89
RF	0.6846	0.7193	0.6923	0.7312	1.12	**1.65**
LSTM Uni	0.8150	0.8247	**0.8187**	**0.8275**	0.45	0.34
LSTM Bi	0.8070	0.8024	0.8110	0.8133	0.50	1.36

Table 6. Results after lowercasing.

Algorithm	Without Pre-Processing		Spelling Corrections		Accuracy improvement (%)	F1-score improvement (%)
	Accuracy	F1-score	Accuracy	F1-score		
SGD	0.7654	0.7733	0.7901	0.7939	**3.23**	**2.66**
SVM	0.7901	0.7941	0.7983	0.8051	1.04	1.39
RF	0.6846	0.7193	0.6983	0.7382	2.00	2.63
LSTM Uni	0.8150	0.8247	**0.8193**	**0.8289**	0.53	0.51
LSTM Bi	0.8070	0.8024	0.8188	0.8152	1.46	1.60

Table 7. Results after spelling corrections.

4.6. Effect of Replacing Slang With Proper Words

Table 8 shows the results obtained after replacing common Swahili slang with proper Swahili words. The Swahili dataset used show that the word "mvulana" can be commonly written in five different ways which are "aguy", "chali", "jamaa", "mhi" and "kijanaa". Without replacing slang, all these six words will be represented with six different vectors. So correcting slang leads to a reduction in the number of vectors generated during vectorization and 0.05% of words are affected by this step. Results show that replacing slang with proper words has a positive effect on the accuracy and f-score for all the trained models. The improvements vary from model to model with SGD registering the highest improvement in accuracy of 2.05% and LSTM Bi achieving the highest improvement in f1-score of 2.23%.

4.7. Effect of Removing Stop Words

Table 9 shows the results obtained after removing common Swahili stop words. 15.67% of all words are affected by this step, which leads to a reduction in the number of vectors to be used during training, hence improving the overall performance. Results show that removing stop words has a positive effect on the accuracy and f-score for all the trained models. The improvements vary from model to model with SGD and RF registering the highest improvements in both accuracy and f1-score. The highest improvement in accuracy is attained by SGD with 6.62%, followed by RF with 6.38%, whereas the highest improvement in f1-score is attained by RF with 8.19%, followed by SGD with 6.01%.

4.8. Effect of Performing All Pre-Processing Steps

Table 10 shows the results obtained after performing all the pre-processing steps, starting with removing punctuations, then lowercasing, then replacing typos with proper Swahili words, then replacing slang with proper Swahili words, and finally removing common Swahili stop words. Results show that performing data pre-processing in that order has a positive effect

Algorithm	Without Pre-Processing		Slang Correction		Accuracy improvement (%)	F1-score improvement (%)
	Accuracy	F1-score	Accuracy	F1-score		
SGD	0.7654	0.7733	0.7811	0.7861	**2.05**	1.66
SVM	0.7901	0.7941	0.8022	0.8085	1.53	1.81
RF	0.6846	0.7193	0.6971	0.7342	1.83	2.07
LSTM Uni	0.8150	0.8247	0.8156	**0.8255**	0.07	0.10
LSTM Bi	0.8070	0.8024	**0.8189**	0.8203	1.47	**2.23**

Table 8. Results after correcting slang.

Algorithm	Without Pre-Processing		Stop Word Removal		Accuracy improvement (%)	F1-score improvement (%)
	Accuracy	F1-score	Accuracy	F1-score		
SGD	0.7654	0.7733	0.8161	0.8198	**6.62**	6.01
SVM	0.7901	0.7941	0.8183	0.8251	3.57	3.90
RF	0.6846	0.7193	0.7283	0.7782	6.38	**8.19**
LSTM Uni	0.8150	0.8247	**0.8332**	**0.8414**	2.23	2.02
LSTM Bi	0.8070	0.8024	0.8275	0.8287	2.54	3.28

Table 9. Results after removing Swahili stop words.

Algorithm	Without Pre-Processing		Performing All Steps		Accuracy improvement (%)	F1-score improvement (%)
	Accuracy	F1-score	Accuracy	F1-score		
SGD	0.7654	0.7733	0.8188	0.8218	6.98	6.27
SVM	0.7901	0.7941	0.8192	0.8284	3.68	4.32
RF	0.6846	0.7193	0.7363	0.7889	**7.55**	**9.68**
LSTM Uni	0.8150	0.8247	**0.8379**	**0.8476**	2.77	2.78
LSTM Bi	0.8070	0.8024	0.8363	0.8375	3.63	4.37

Table 10. Results after performing all pre-processing steps.

on the accuracy and f-score for all the trained models. The improvements vary from model to model with RF registering the highest improvement in accuracy and f1-score of 7.55% and 9.68% respectively. Note that this is the highest improvement as compared to the previous results which only showed improvements after applying one of the steps.

4.9. Visual Comparison of Average Performance Impact

Fig. 2 shows a visual comparison of performance improvements in percentages of the f1-score and accuracy by using Doc2Vec as the text representation technique. The graphs show that all the tested algorithms are positively affected by the application of pre-processing steps. Generally, the highest average impact for the experimented pre-processing steps is attained by RF followed by SGD for both accuracy and f1-score metrics.

4.10. Visual Comparison of the Impact of Pre-Processing

Fig. 3 and Fig. 4 show a visual comparison of the accuracy and f1-score improvement in percentages for each of the tested algorithms. The graphs show a positive effect for all the models. Generally, RF and SGD are highly affected, while LSTM-Uni and SVM are the least affected by pre-processing for both accuracy and f1-score. The pre-processing step of removing stop words has the most impact on both accuracy and f1-score.

4.11. Average Performance of the Tested Models

Table 11 shows the average performance of the tested algorithms in terms of accuracy and f1-score. Generally, we obtain the highest average accuracy and f1-score for LSTM-Uni with 82.35% and 83.29% respectively, followed by LSTM-Bi with an accuracy of 82.02% and

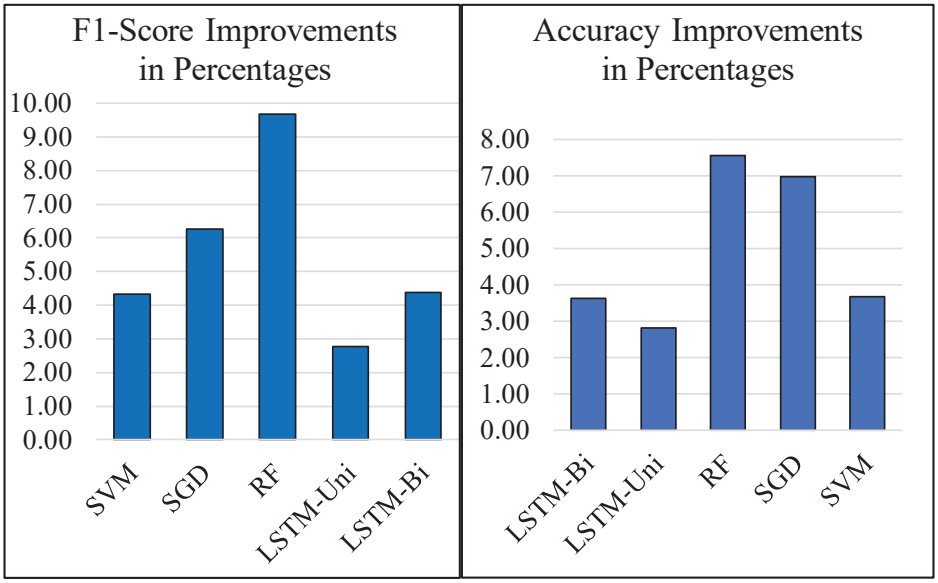


Figure 2. Average model improvements on the performance after applying all pre-processing steps.

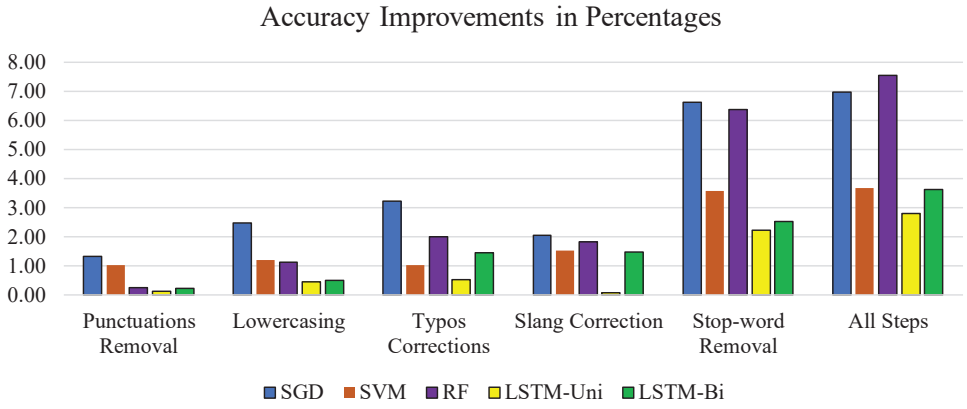


Figure 3. Accuracy improvements from each pre-processing step for the tested models.

f1-score of 82.07% when hyper-parameter tuning is used as a baseline.

4.12. Sequential Improvements of Accuracy and F1-Score

Fig. 5 and Fig. 6 show the sequential improvements of the accuracy and f1-score for each of the tested algorithms after combining the pre-processing steps in the sequence: 1 = hyper-parameter fine-tuning, 2 = removing punctuations, 3 = lowercasing, 4 = correcting typos, 5 = correcting slang, 6 = removing stop words. The graphs show a positive effect for all the tested models with the applied pre-processing sequence. The hyper-parameter fine-tuning step has the most impact on both the accuracy and f1-score for all the models. Also, the combination of all pre-processing steps records the highest impact on improving the performance of the tested models.

5. CONCLUSION

The detailed experiments presented show that all the pre-processing steps when applied separately on Swahili textual data have a positive impact on the performance of all the evaluated classification algorithms. Among the experimented pre-processing steps, removing stop words has the highest impact on the performance of both accuracy and f1-score metrics when hyper-parameter tuning is used as a baseline step. The punctuation removal step has the lowest impact on the performance of both the accuracy and f1-score. Punctuation removal records the lowest impact because the Swahili textual dataset used has a low percentage

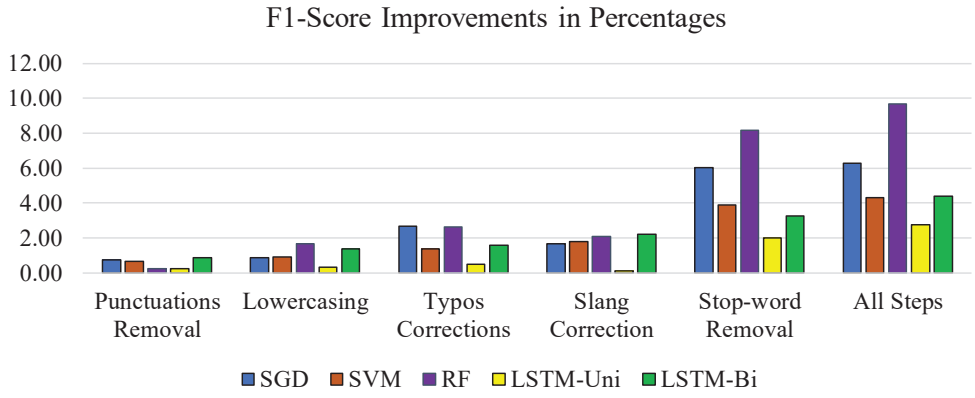


Figure 4. F1-score improvements from each pre-processing step for the tested models.

Algorithm	Average Accuracy Performance	Average F1-Score Performance
SGD	0.7944	0.7968
SVM	0.8060	0.8113
RF	0.7065	0.7486
LSTM-Uni	**0.8235**	**0.8329**
LSTM-Bi	0.8202	0.8207

Table 11. Average performance of the tested models.

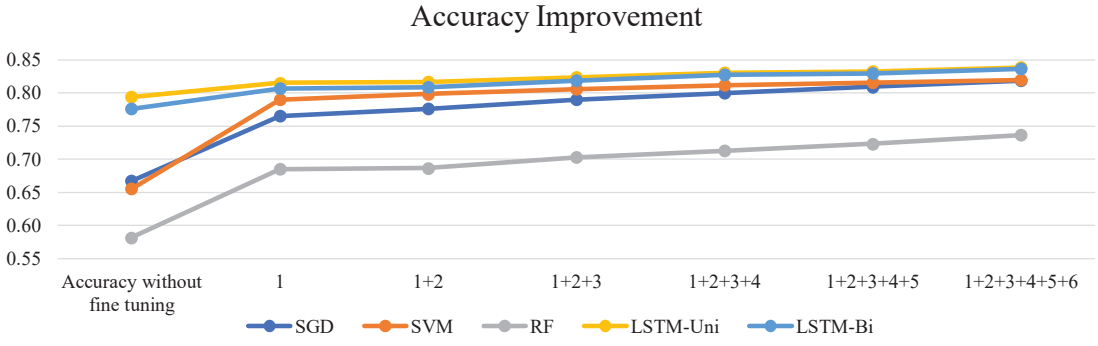


Figure 5. Accuracy improvements after combining the pre-processing steps sequentially.

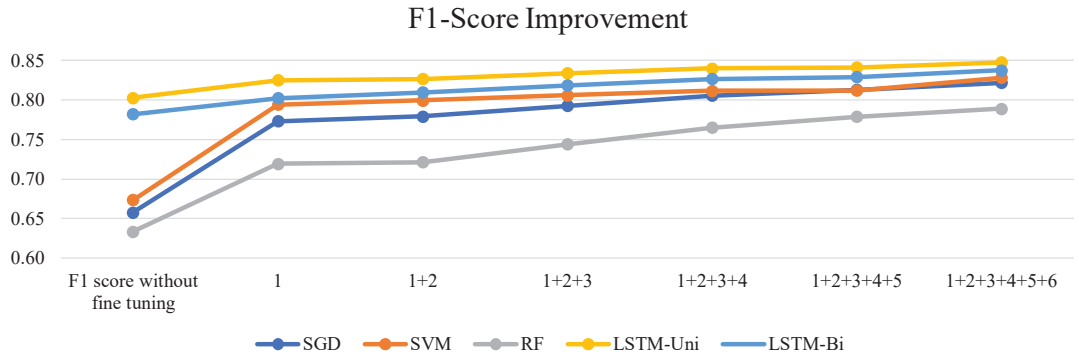


Figure 6. F1-score improvements after combining the pre-processing steps sequentially.

of punctuation characters and the effect may increase when dealing with a dataset containing a larger number of punctuation characters.

Among all the evaluated algorithms, RF and SGD are the most affected by the pre-processing steps on the Swahili textual dataset, whereas LSTM-Uni and SVM are less affected by the pre-processing steps experimented in this study. The study shows that the LSTM-Uni algorithm has the best performance for the classification of Swahili textual data compared to the other tested algorithms.

We believe that our study results will help future NLP and Machine Learning researchers to carefully select these text pre-processing methods when dealing with Swahili textual data. Finally, as future work, we plan to work on improving typos, slang, stop words and Swahili corpus datasets, to accommodate more words that will be applied in different topics and to cover different platforms, both formal and informal platforms, such as social media. We will also add more pre-processing steps, for example, stemming, lemmatization, emoticons replacement, abbreviations and acronyms replacement, and so on, to our experiment and discover the individual impact of each pre-processing step in terms of the performance of machine learning algorithms on Swahili textual datasets.

Conflict of Interest

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this article.

Authors' Contribution

All authors contributed to the study conceptualization and design. Material preparation, data collection and analysis were performed by Bernard Masua and Noel Masasi. The first draft of the manuscript was written by Bernard Masua and Noel Masasi, while Hellen Maziku and Betty Mbwilo commented on prior versions of the manuscript. All authors read and approved the final article.

Bernard Masua: study conceptualization and design; data collection, data curation, data analysis and data visualization; manuscript writing, reviewing and editing.
Noel Masasi: study conceptualization and design; data collection, data curation, data analysis and data visualization; manuscript writing, reviewing and editing.
Hellen Maziku: study conceptualization and design; administration and supervision; manuscript reviewing and editing.
Betty Mbwilo: study conceptualization and design; supervision; manuscript reviewing and editing.

Funding

The authors declare that no funding was obtained for this study.

Data Availability

The common Swahili slang dataset has been published here: https://data.mendeley.com/datasets/b8tc96xf3h/1. Repository name: Mendeley Data, DOI: https://doi.org/10.17632/b8tc96xf3h.1.

The common Swahili typos dataset has been published here: https://data.mendeley.com/datasets/3xmsjhdrc9/1. Repository name: Mendeley Data, DOI: https://doi.org/10.17632/3xmsjhdrc9.1.

The common Swahili stop-words dataset has been published here: https://data.mendeley.com/datasets/mmf4hnsm2n/1. Repository name: Mendeley Data, DOI: https://doi.org/10.17632/mmf4hnsm2n.1.

Source codes are available and can be accessed through a public GitHub repository here: https://github.com/LeoVinciTZ/Swahili-NLP/blob/master/Text%20Classification/Topic_Classification.ipynb.

Ethics Approval/Consent

Research clearance letters were issued to involved organizations for data collection. During the data collection, respondents were briefed on the benefits of the study and were asked to participate voluntarily. Respondents were informed and assured that the research was part of academic requirements.

REFERENCES

[1] C.S. Shikali, Z. Sijie, L. Qihe, R. Mokhosi. Better Word Representation Vectors Using Syllabic Alphabet: A Case Study of Swahili. Applied Sciences, Vol. 9(18), p. 3648, 2019.

[2] R. Řehůřek, P. Sojka. Software Framework for Topic Modelling with Large Corpora. Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Malta, pp. 46–50, 2010.

[3] N. Masasi, B. Masua. Common Swahili Stop-Words. Mendeley Data, 2020.

[4] N. Masasi, B. Masua. Common Swahili Slangs. Mendeley Data, 2020.

[5] N. Masasi, B. Masua. Common Swahili Typos. Mendeley Data, 2020.

[6] W. Etaiwi, G. Naymat. The Impact of Applying Different Preprocessing Steps on Review Spam Detection. Procedia Computer Science, Vol. 113, pp. 273–279, 2017.

[7] M. Ott, C. Cardie, J.T. Hancock. Negative Deceptive Opinion Spam. Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013), Atlanta, Georgia, pp. 497–501, 2013.

[8] M. Ott, Y. Choi, C. Cardie, J.T. Hancock. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2011), Portland, Oregon, USA, pp. 309–319, 2011.

[9] M. Işik, H. Dağ. The Impact of Text Preprocessing on the Prediction of Review Ratings. Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 28(3), pp. 1405–1421, 2020.

[10] S. Alam, N. Yao. The Impact of Preprocessing Steps on the Accuracy of Machine Learning Algorithms in Sentiment Analysis. Computational and Mathematical Organization Theory, Vol. 25(3), pp. 319–335, 2019.

[11] S. Symeonidis, D. Effrosynidis, A. Arampatzis. A Comparative Evaluation of Pre-Processing Techniques and Their Interactions for Twitter Sentiment Analysis. Expert Systems With Applications, Vol. 110, pp. 298–310, 2018.

[12] N.A. Smith. Contextual Word Representations: Putting Words Into Computers. Communications of the ACM, Vol. 63(6), pp. 66–74, 2020.

[13] J. Aguilar, C. Salazar, H. Velasco, J. Monsalve-Pulido, E. Montoya. Comparison and Evaluation of Different Methods for the Feature Extraction from Educational Contents. Computation, Vol. 8(2), p. 30, 2020.

[14] L. Breiman. Random Forests. Machine Learning, Vol. 45(1), pp. 5–32, 2001.

[15] K. Kirasich, T. Smith, B. Sadler. Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets. SMU Data Science Review, Vol. 1(3), p. 9, 2018.

[16] L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In: Y. Lechevallier, G. Saporta (eds.), Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT 2010, Paris), Physica-Verlag HD, New York, pp. 177–186, 2010.

[17] I. Loshchilov, F. Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. Poster at the 5th International Conference on Learning Representations (ICLR 2017), 2017.

[18] J.L. Elman. Finding Structure in Time. Cognitive Science, Vol. 14(2), pp. 179–211, 1990.

[19] H. Jelodar, Y. Wang, R. Orji, S. Huang. Deep Sentiment Classification and Topic Discovery on Novel Coronavirus or COVID-19 Online Discussions: NLP Using LSTM Recurrent Neural Network Approach. IEEE Journal of Biomedical and Health Informatics, Vol. 24(10), pp. 2733–2742, 2020.

[20] A. Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. Physica D: Nonlinear Phenomena, Vol. 404, p. 132306, 2020.

[21] C. Cortes, V. Vapnik. Support-Vector Networks. Machine Learning, Vol. 20(3), pp. 273–297, 1995.

[22] D.A. Pisner, D.M. Schnyer. Chapter 6 - Support Vector Machine. In: A. Mechelli, S. Vieira (eds.), Machine Learning: Methods and Applications to Brain Disorders. Academic Press, London, UK, pp. 101–121, 2020.

[23] K.M. Ting. Confusion Matrix. In: C. Sammut, G.I. Webb (eds.), Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA, p. 260, 2017.

[24] R.G. Mantovani, T. Horváth, R. Cerri, J. Vanschoren, A.C.P.L.F. de Carvalho. Hyper-Parameter Tuning of a Decision Tree Induction Algorithm. Proceedings of the 5th Brazilian Conference on Intelligent Systems (BRACIS 2016), Recife, Brazil, pp. 37–42, 2016.

[25] R.G. Mantovani, A.L.D. Rossi, J. Vanschoren, B. Bischl, A.C.P.L.F. de Carvalho. Effectiveness of Random Search in SVM Hyper-Parameter Tuning. Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, pp. 1–8, 2015.

[26] P. Lameski, E. Zdravevski, R. Mingov, A. Kulakov. SVM Parameter Tuning with Grid Search and Its Impact on Reduction of Model Over-Fitting. In: Y. Yao, Q. Hu, H. Yu, J.W. Grzymala-Busse (eds.), Proceedings of the 15th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing (RSFDGrC 2015). Lecture Notes in Computer Science, Springer, Cham, Vol. 9437, pp. 464–474, 2015.

[27] B. Masua, N. Masasi. GitHub Repository. 2021. Available Online: https://github.com/LeoVinciTZ/Swahili-NLP/tree/master/Text Classification/Models.