LONTAR KOMPUTER VOL. 13, NO. 1 APRIL 2022 **p-ISSN 2088-1541**
DOI : 10.24843/LKJITI.2022.v13.i01.p05 **e-ISSN 2541-5832**
Accredited Sinta 2 by RISTEKDIKTI Decree No. 158/E/KPT/2021

Spam Comments Detection on Instagram Using Machine Learning and Deep Learning Methods

Antonius Rachmat Chrismanto[a1], Afiahayati[b2], Yunita Sari[b3], Anny Kartika Sari[b4], Yohanes Suyanto[b5]

[a]Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana
Yogyakarta, Indonesia
[1]anton@ti.ukdw.ac.id (Corresponding author)
[2]afia@ugm.ac.id

[b]Department Computer Science and Electronics, Universitas Gadjah Mada
Yogyakarta, Indonesia
[3]yunita.sari@ugm.ac.id
[4]a_kartikasari@ugm.ac.id
[5]yanto@ugm.ac.id

Abstract

The more popular a public figure on Instagram (IG), the number of followers also increase. When a public figure posts something, there are many comments from other users. In fact, from all the comments, not all of them are relevant to the post, such as advertising, links, or clickbait comments. The type of comments that are irrelevant to the post is usually called spam comments. Spam comments will interfere with information flow and may lead to misleading information. This research compares machine learning (ML) and deep learning (DL) classification methods based on our collected Indonesian IG spam comment dataset. This research was conducted in the following steps: dataset preparation, pre-processing, simple normalization, features generation using TF-IDF and word embedding, application of ML and DL classification methods, performance evaluation, and comparison. The authors compare accuracy, F-1, precision, and recall from ML and DL results. This research shows that ML and DL methods do not significantly differ. The Linear SVM, Extreme Tree (ET), Regression, and Stochastics Gradient Descent algorithms can reach the accuracy of 0.93. At the same time, the DL method has the highest accuracy of 0.94 using the SimpleTransformer BERT architecture. The difference between ML and DL methods is not significantly different.

Keywords: *Spam Comments Detection, Instagram (IG), Deep Learning, Machine Learning*

1. Introduction

Social media allows users to carry out various activities like society's real world. Social media will enable users to relate, make friends, convey ideas, convey aspirations, comment on each other, collaborate, and more. Social media users can also transact, raise funds, follow and be followed, do promotions/campaigns, and many other things. Social media is becoming very popular today because of these advanced features.

Facebook (FB), YouTube (YT), Tik-Tok (TK), Instagram (IG), and Twitter (TW) are some popular social media used globally and in Indonesia [1]. These social media have many registered users with advanced features. Public figures, such as politicians, actors, and artists, use these social media to increase their popularity on the Internet. Actors and artists (from now on called artists) usually use social media to promote their activities, increase their popularity, interact with their fans, and many more. The more popular some artists are, the more followers they have. The more followers they have, the more content can also be delivered.

TW, YT, and IG are popular social media that contain much spam. Both are spammer or content, and these social media are also widely used for spam detection research. The more famous an artist is, the higher the spam content of comments on each post [2][3]. The more famous the artist and the more followers they have, the more likely they have spam comments on their social media accounts [2]. Spam comments disrupt information flow from a particular post/status information [4]. This paper uses the study case in IG because of its vast spam comments data. The posting and comments on IG have the following characteristics: using informal language, using lots of emoticons/emojis, lots of abbreviations and many typos, using a lot of code-mix data language, and they have varying lengths of comments long (usually 1-3 sentences, and in 1 sentence consists of five words). IG also has a reply-response structure and no hierarchy and only can use mentions with @ sign [5].

Solutions to deal with spam comments in IG can be done in some techniques, but most are done manually. IG users can manually delete the spam comments, which takes much time and should be checked. IG also provides a feature to report a comment as spam manually, but it must be done one by one. Another solution to minimize spam comments is to change the IG account to private. Making the IG private is difficult for artist / public figure accounts because if they make their IG account private, the other users cannot immediately follow them. The final solution is to set and activate the IG feature to automatically block comments containing certain words (the users must enter those keywords by themself, which they consider spam). Blocking spam comments using keywords has disadvantages because they can only be used in a few languages, such as English, and cannot be applied in Indonesian until now [6].

Researchers have researched spam comments detection in IG previously, most in English and some in other languages, including Indonesia [7], [8], [9], [10], [11], [5]. In the previous research, the authors tried to develop a spam comment detection service on IG based on the REST web service [12]. The authors implemented it in a Firefox extension by ordinary users in the real world [13]. The actual implementation using browser extensions proves that the accuracy results are not good. There are several reasons related to the low accuracy such as, 1) the IG comment is very unstructured and even abnormal, 2) many comments have a lot of symbols and emojis/emoticons, 3) there are many typos or uncommon abbreviations, slang words, and also code-mixing (mixed languages), 4) some comments are deliberately disguised not to be detected by spam, such as the use of the “\ /” characters to write the letter ‘V’ which cause the system cannot read the original character, and 5) the system cannot know the meaning/semantics information of the relationship between a post and comments on IG. Another problem is some posts have only images/pictures and no text caption at all. Spam detection in social media actually are a vast research area and require a lot of solving methods that support one another.

Various machine learning methods, combined with natural language processing (NLP), can be used as social media spam comment detection techniques. In the article [14], there are 11 best detection methods in the classification used and compared, namely Gradient Boosting Trees (GBDT), Random Forest (RF), Extreme Learning Machine (ELM), Support Vector Machine (SVM), C.45, Sparse Representation based Classification (SRC), KNN, Logistic Regression (LR), AdaBoost (AB), Naïve Bayes (NB), and Feed Forward Neural Network (FFNN). Article [14] studies indicate that GBDT has almost the same performance, exceeds SVM and RF's prediction performance, and is the fastest algorithm in time efficiency during prediction. ELM, GBDT, RF, SVM, and C4.5 have adequate accuracy, but this performance varies widely across all datasets. The FFNN method has the worst accuracy but the second-fastest prediction efficiency after GBDT. SRC shows good accuracy performance but is the slowest in training and testing [14].

Deep learning has recently used well-known methods such as Convolutional Neural Network (CNN) for image classification and RNN (LSTM) for text classification. For example, CNN was used in signature detection and gave an excellent accuracy of 99.4% [15]. CNN is also used in EEG to detect excessive daytime sleepiness and get a good answer [16]. In contrast to CNN, widely used for image processing, spam detection uses a lot of RNN and its variants. Spam detection appears on SMS [17], which uses the SMS UCI dataset using CNN based on hand-engineered features. SMS spam detection was also performed using LSTM [18], [19] RNN-LSTMand, then compared with machine learning methods. Spam in social media can appear as spammer and spam content. Spam content appears on social media, such as TW [20], FB, and IG.

Chrismanto et al.; researchers detect spam posts by spammers on IG using the English language [6]. They use the Random Forest (RF) machine learning method to prioritize the special hand-engineered features on a dataset of 1983 posts content and 953808 media posts. The hand-engineered features are: whether or not mentioned to other accounts, the number of hashtags, the use of hashtags unrelated to the post, repeated words, specific spam keywords defined by the researcher, and the post image that contains a watermark or not. Based on these hand-engineered features, the detection results are relatively high, reaching 96.27% with k-fold validation k = 10. The weakness of [6] is using hand-engineered features that require human labor for the extraction.

Research [21] differs from [7] because it uses the Indonesian language, not English, and detects spam posts but spam comments. The dataset comes from the dataset of Indonesian public figures. Unlike this research, spam comments in [21] have appropriate promotional objectives (such as advertising merchandise). The features used are a combination of hand-engineered features, keyword features, and text features themselves. The hand-engineered features are the comment's length, the number of capital letters, and emojis. The text features used are 1) Bag of Words (BOW), 2) TF-IDF, and 3) Fasttext, which is used in various combinations. The classification methods used are NB, SVM, and XGBoost. The result was that using all features (1, 2, and 3) resulted in an F1 of 0.96. This study states that the features used are highly dependent on the dataset and cannot be generalized to all other new data, especially for keywords retrieved using regular expressions.

Not much research has been done on detecting spam comments on IG in the Indonesian language. NB algorithm uses the classification method and has an accuracy of 72% [8]. In contrast, the opposite NB algorithm, namely Complementary Naïve Bayes (CNB) in an unbalanced dataset (non-spam comments outnumber spam comments) between spam and non-spam comments, has better accuracy [9]. The CNB algorithm can produce 92% accuracy, while the SVM algorithm gets 87%.

The pre-processing technique is almost identical to various studies using text data for the classification problem, including the IG dataset. All pre-processing techniques used to detect spam posts or comments must be done using the NLP method. The pre-processing text has a significant impact before the further stage, features generation and selection [22]–[24]. The pre-processing technique uses tokenization or n-gram tokenization (split sentence into words), case-folding, stop words removal, post tagging, normalization, spelling correction, and stemming. The least effective pre-processing technique is stemming [24].

The authors have also conducted various studies related to this topic before. The first research on the collection of the 2017 IG dataset and the use of the NB [6], SVM [10], KNN [4], and DW-KNN [11] used the RapidMiner and PHP. The best two methods based on our previous studies are the KNN and DW-KNN. However, the accuracy is still not good enough and can still be improved using appropriate deep learning methods.

This research tries to compare the performance of some machine learning based on [14] and several deep learning methods on the IG comments dataset obtained from 10 artists with more than 10 million followers [25]. This research aims to contribute experimental results and comparisons of accuracy, precision, recall, and F-1 on the Indonesian IG spam comments dataset. To our knowledge, this comparison from the IG spam in the Indonesian language case study has not been made before. These results will be the initial part of conducting more in-depth research and analysis to improve detection and search for gaps/improvements in detecting spam comments on IG with various unique characteristics. This research wants to contribute to get the comparison performance of some ML and DL algorithms.

2. The Research Methods

The primary process of this research is carried out in five major steps. The research methodology of this research is carried out as follows: 1) Data gathering, 2) Data cleaning, pre-processing, and normalization, 3) Implementation of spam comment detection using machine learning, 4) Implementation of spam comment detection algorithm using deep learning, and 5) Comparison results, discussions, and analysis. The details of each step will be described more clearly in the following subsection.

LONTAR KOMPUTER VOL. 13, NO. 1 APRIL 2022 p-ISSN 2088-1541
DOI : 10.24843/LKJITI.2022.v13.i01.p05 e-ISSN 2541-5832
Accredited Sinta 2 by RISTEKDIKTI Decree No. 158/E/KPT/2021

2.1. Data Gathering

The primary data source is obtained from Instagram. Dataset is built from Indonesian artists' postings, and comments in the Indonesian language, with more than 10 million followers collected in 2017 [6]. The profile of the dataset before cleaning can be seen in Table 1.

Table 1. Instagram Annotated Dataset Before Cleaning, Preprocessing, and Normalization

No.	Artist	Class name and total
1.	Ayu Tingting	Spam (1262), Not Spam (584)
2.	Julia Perez	Spam (1362), Not Spam (739)
3.	Nagita Slavina	Spam (1435), Not Spam (610)
4.	Syahrini	Spam (922), Not Spam (448)
5.	Laudya Cinthia Bella	Spam (902), Not Spam (688)
6.	Prili Latuconsina	Spam (437), Not Spam (1091)
7.	Chelsea Olivia	Spam (1625), Not Spam (293)
8.	Luna Maya	Spam (965), Not Spam (275)
9.	Raisa	Spam (666), Not Spam (621)
10.	Agnes Monica	Spam (1143), Not Spam (940)
Total Spam 10.719		**Total Not Spam 6.288**
General Total 17.007		

The dataset can be accessed from https://ig-repo.fti.ukdw.ac.id/ in JSON, XML, or plain text (non-Unicode) format. The dataset profile consists of not-spam of 5187 and spam of 9313. The total number of data is 14500 data. The dataset's characteristics are 1) it consists of duplicate letters and punctuation, 2) there are a lot of Unicode symbols, 3) it contains emoticons/emoji, 4) there are a lot of non-standard abbreviations, 5) it has lots of misspelled words (typo), 6) it contains custom symbols, and 7) it contains code-mixing language (a mixture of Indonesian and others).

2.2. Data Cleaning, Pre-Processing, And Normalization

This research used all the datasets for the experiment. From 14500 data, it splits into data training and testing using Pareto 80:20. The amount of training data used is 11600, 2900 data for data testing with K-Fold validation in ML method and random split in DL method. The data pre-processing and normalization step are carried out to clean and prepare the dataset for the next step. The pre-processing steps are 1) case folding, 2) tokenization, 3) punctuation removal, 4) emoji removal, 5) double characters removal (etc.: sayaaaa!! (in English: me), cobaa…. (in English: try)), 6) Stop words removal, and 7) Stemming using Python Sastrawi library. In the case of folding, all comments will be changed to lowercase letters. Tokenization will break sentence text into word tokens. The next step is removing all punctuation marks and normalizing each word's letters. Emoji also be removed from the data because emoji is categorized as a symbol, not text. Stop word removal means removing words that are not important using the stop word list. Stemming changes the word into a root word if it has not in the form of a root word—stemming aims to reduce the number of tokens that appear.

Simple normalization is also done to reduce typos (writing errors). Some techniques can be used to overcome the writing error (typos), such as dictionary-based, edit distance, similarity key, rule-based, and probabilistic [43],[44]. In this research, the authors use some normalization steps modified from [45] done as follows:

1. The system accepts dataset input in CSV format.
2. The system loads the KBBI dictionary into memory.
3. The system loads the abbreviation dictionary into memory.
4. The system prepares results in a txt file for the final normalized dataset and an evaluation.txt file to store tokens that were replaced due to spelling correction to evaluate their accuracy.
5. For each row in the dataset, pre-processing is carried out, such as lowering the case, entering normalization, and normalization of adjacent twin letters to be reduced to 2 letters

6. The system prepares a modified Peter Norvig-based spelling correction function using a word embedding from Wikipedia.
7. The system then tokenizes sentences on the dataset with the NLTK library.
8. For each word/token in the tokenization results, the following process is carried out:
    a) Handling for the letter x representing the string "katax" ("katanya"). For example, katax is 'they said" in English,
    b) handling for letter 2, which describes repeated words. For example, 'kupu2' = butterflies,
    c) Check if the token is in the KBBI dictionary. If there is, then it is considered valid and then saved to the result_token
    d) If it is not in the KBBI, then the subsequent examination is continued, namely the examination of the abbreviation / 'alay' dictionary, for the non-standard words dictionary, e) If there is, then save it to the result_token,
    e) If there is none, then the results will be published and proceed to stem,
    f) If the stemming result is different from the previous input token, then proceed to the spelling correction process using the Peter Norvig algorithm based on Wikipedia's Word2Vec,
    g) If the result of the spelling correction is still the same as the previous token, which means the correction was not successful, then it is stored in the result_token as is,
    h) If the correction result differs from the previous token, there is a correction process. The result is rechecked in the KBBI dictionary,
    i) If it is found in the KBBI dictionary and the word class is the same as the original token, the correction process was successful, and the result_token = results_final_correction. The results are also recorded in the evaluation.txt file, and,
    j) The token will be left as if it is not in the KBBI dictionary.
9. For one row of a processed dataset, the result is stored in a Python list to be used later (result.txt file) per row until all rows are processed in the dataset.
10. Finally, the results.txt and evaluation.txt files are saved on the hard disk by Python, and the process is declared complete.

2.3. Implementation of The Machine Learning Algorithm

Machine learning methods consist of two types: supervised learning and unsupervised learning. Detection/classification problems are included in the supervised learning category, although some references also said it could be done with semi-supervised or weak supervised models. The weakly supervised method uses the concept that by using a few labels on the dataset, the classification process can be done by using learning outcomes with a few labels to classify/label other data that have not been labeled [26], [27].

This research uses the methods used in the article [14] (NB, SVM, KNN, AdaBoost, DT, RF, XGBoost, and LR methods). It will compare each performance in IG's spam comments detection case with the IG dataset. For the first method, Naïve Bayes (NB) is based on the Bayes theorem with the naïve assumption of each feature pair in each class [28]. Bayes' theorem where y is class and x1 to xn can be formulated in Formula (1).

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)} \qquad (1)$$

Assuming the naïve condition is free as in Formula (2).

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y), \qquad (2)$$

NB will predict whether x is categorized as class y based on all data, which has the highest posterior probability as in Formula (3).

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \dots, x_n)} \qquad (3)$$

As can be seen in Formula (3), $P(x_1, \dots, x_n)$ It is constant, so Formula (3) can be simplified into

Formula (5) and Formula (6).

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \quad (4)$$

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y) \quad (5)$$

The SVM method is a well-known method that is very good in classifying two (binary) classes. It is still valid when the data dimensions are high, including when the number of dimensions is greater than the number of samples. It is memory efficient and has many kernel tricks used in various cases [29]. The SVM algorithm is a classifier algorithm based on Vapnik's supervised learning model in 1992. The SVM method will search and find a hyperplane and measure x-1 dimensions to separate training data based on categories or classes in several training data with several x attributes (the vector has a size of x dimensions), t. Finding a hyperplane is done by maximizing the distance between classes (margin). SVM can guarantee high generalizability for future data [30].

The SVM will calculate the optimization problem with Formula (6) [31]. Suppose the training data is the data that has been labeled and has several x attributes (commonly known as tuples) (xi, yi). Where i = 1, 2,…, n, n is the amount of training data, xi is the set of attributes in the ith, and yi is the class of the ith training data.

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{x} \xi_i \quad (6)$$

By the provisions in accordance with Formula (7).

$$y_i(w^T \phi(x_i)) + b \geq 1 - \xi_i \text{ dan } \xi_i > 0 \quad (7)$$

KNN is a supervised learning method where the new data are classified based on most k-nearest neighbor categories. The KNN algorithm uses neighborhood classification to predict new data. KNN for text classification can be seen in [32] with an average accuracy level, reaching 95%. The principle of the k-NN is to find the closest distance between the data to be evaluated and the closest neighbors in the training data, where k is the number of closest neighbors. The steps in the KNN algorithm are 1). determine k, 2). calculate the distance (similarity) between new and other data, 3). sort the distance by the threshold k, and 4). use the closest distance to most class members. The distance formula can be seen in Formula (8).

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (8)$$

Gradient boosting algorithms are used for regression and classification problems. There are three gradient boosting elements: the weak function, weak learner, and adaptive model. The loss function is highly dependent on the dataset, the weak learner can make predictions, and the additive model minimizes the loss function by adding a weak learner. The Ada Boost (Adaptive Boosting) algorithm is a meta-algorithm that tries the classifier on the original dataset and then adjusts it from the classifier on the same dataset. The incorrectly classified data's weight is adjusted again so the next classifier can classify it better [33]. Another boosting algorithm is also found in the XGBoost algorithm (eXtreme Gradient Boosting). This algorithm combines models with low accuracy to create models with higher accuracy. XGBoost is based on a decision tree developed by Tianqi Chen. Because XGBoost was born from a library, its development is implemented in many programming languages such as C ++, Python, R, Julia, and Java. The models supported by XGBoost are the regular Gradient Boosting model, SGD (Stochastic Gradient Boosting), and Regularized Gradient Boosting with L1 and L2 regularization [34].

The Extra Tree (ET) algorithm is also developed based on a decision tree and an ensemble with a random forest in its development. The extra tree classifier, such as RF, makes arbitrary decisions and randomizes specific data subsets to minimize over-fitting and over-learning [35]. Some parameters that can be changed are the number of trees, features, and the minimum sample per split [36].

Some best machine learning classification methods are applied to process datasets at this stage. The machine learning algorithms used are 1) NB, 2) CNN, 3) Linear SVM, 4) SVM Radial Basis Function (RBF), 5) KNN, 6) Ada Boost, 7) DT, 8) RF, 9) eXtreme Gradient Boosting (XGBoost), 10) Stochastic Gradient Descent (SGD), 11) Extra tree and 12) Multi-Layer Perceptron (MLP). The methods are implemented using Python and Scikit-learn Library.

2.4. Implementation Of The Deep Learning Algorithm

This research also uses suitable deep learning methods for sequential text data processing and shallow learning machine learning. The deep learning methods used in this research are Recurrent Neural Network (RNN), Long-short Term Memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional LSTM (Bi-LSTM), and Transformers (using SimpleTransformers library).
The text classification process using deep learning accepts input in a word embedding. Word embedding can be in the form of Word2Vec by Google [37], Fasttext by Facebook [38], or Glove [39]; after the word embedding is created through the training process from the dataset itself, the word embedding is used as input from the classification architecture layer.

RNN architecture is a deep learning architecture that processes sequential data based on time steps. RNN is suitable for text data processing or other time series such as forecasting/prediction [40]. Especially for text data, The RNN architecture can process sequences of sentences by word/token processing. RNNs have several drawbacks, such as 1) the operations are sequential, so the processing cannot be done in parallel, 2) vanishing gradient may occur, and 3) the training process takes too long [39].

LSTM architecture tries to overcome the weaknesses in RNN in terms of vanishing gradient [41]. LSTM is usually used in text processing and time series data [42] for predicting sea level. LSTM uses different gates in its architecture, consisting of an input gate, a forget gate and an output gate. The LSTM architecture does not cause vanishing Gradient and makes the system forget less important information. Some LSTM variant are GRU (Gated Recurrent Unit) [43] and Bi-LSTM [44]. A Bi-LSTM is an LSTM that uses two layers of LSTM, one that receives input in the forward direction and the other in the reverse direction. Bi-LSTM effectively increases the information available to the network and the processing context. GRU is an extension of the standard LSTM with some modification gates. GRU has two gates (reset and upgrade gates), while the LSTM has three gates (input, output, and forget ports).

RNN, LSTM, GRU, and Bi-LSTM, previously discussed, still have weaknesses such as 1) they cannot be processed in parallel, 2) there is always a chance that a vanishing gradient will occur, and 3) the training process is slower [45]. Google Brain created a new architecture called Transformer to overcome the previous problems. Transformer architecture relies only on the attention mechanism [46]. LSTM makes training faster, has no vanishing gradients, and the process can be done in parallel. Transformer achieves state-of-the-art (SOTA) in Neural Machine Translation (NMT) processing [45].

2.3. Performance Evaluation

The last stage of this research is performance evaluation to see the performance comparison between ML and DL methods in spam comments detection based on the IG dataset. The evaluation matrix used in this experiment is the confusion matrix, as shown in Table 2. A confusion matrix (CM) is a simple matrix to evaluate classification performance by machine/computer. CM is a table with a minimum of 4 different combinations of predicted values by machine and the actual values. It supports binary or more classification.

Tabel 2. Confusion Matrix

		Predicted Class	
		Negative	**Positive**
	Negative	True Negative (TN)	False Negative (FN)

Actual Class	**Positive**	False Positive (FP)	True Positive (TP)

Where:

- True-negative = the number of negative data that is correctly categorized as a negative class
- False-negative = the number of negative data that is categorized as the positive class
- False-positive = the number of positive data that is categorized as a negative class
- True-positive = the number of positive data that is true that is categorized as a positive class

Further calculations can be carried out from the confusion matrix in Table 2 to get accuracy, recall, precision, and F-measure with Formula (8) to Formula (11).

$$Accuracy = \frac{TN+TP}{TN+FP+FN+TP} \tag{9}$$

$$\text{Recall or Sensitivity} = \frac{TP}{(FP+TP)} \tag{10}$$

$$Precision = \frac{TP}{(FN+TP)} \tag{11}$$

$$F1\ Score = \frac{2*TP}{(2*TP+FP+FN)} \tag{12}$$

3. Result and Discussion

3.1. Result of the Machine Learning Methods

This experiment is done using some configurations, as seen in Table 3. For the ML methods, every method runs the script in batch mode. The dataset is split into train and test in 80:20. ML features using TF-IDF normalize vectors with a 1-gram token. This ML experiment shows that the best performance is achieved by linear SVM, linear Regression, SGD, and extra tree in both accuracy and F1 score of 0.93 (see details in Table 4).

Table 3. Table. The Machine Learning Parameters

Experiment Parameter	**Value**
Python libraries	TensorFlow, sci-kit-learn, pandas, NumPy, matplotlib, seaborn, gensim, tqdm, simpletransformers, nltk, string, itertools, xgboost
MultinomialNB, ComplementNB()	default
LinearSVC()	random_state=42, tol=1e-5
SVM()	C=1.0, gamma='auto'
KNN	n_neighbors=3
AdaBoost	n_estimators=100, random_state=42
DecisionTree	random_state=42
RandomForest	max_depth=2, random_state=42
Logistics Regression	multi_class='multinomial',solver='saga', max_iter=100
eXtreme Gradient Boosting	objective='binary:hinge'
Stochastic Gradient Descent	max_iter=1000, tol=1e-3
ExtraTree	n_estimators=100, random_state=42
Multi Layer Perceptron	random_state=42, max_iter=300

Table 4. Machine Learning Methods Performance Result

Method / Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**
NB	0.91	0.89	0.89	0.89
CNB	0.92	0.89	0.92	0.90
Linear SVM	**0.93**	0.92	0.93	**0.93**
SVM Radial Basis Function	0.64	0.32	0.50	0.39
KNN	0.82	0.85	0.76	0.78
AdaBoost	0.90	0.89	0.92	0.9
Decision Tree	0.91	0.9	0.92	0.91

RF	0.64	0.32	0.5	0.39
Linear Regression	**0.93**	0.92	0.94	**0.93**
XGBBoost	0.89	0.88	0.91	0/89
SGD	**0.93**	0.92	0.94	**0.93**
Extreme Tree	**0.93**	0.92	0.93	**0.93**
Multi Layer Perceptron	0.91	0.9	0.9	0.9

3.3. Result of the Deep Learning Methods

In the deep learning method's evaluation, we use some scenarios, as seen in Table 5. The scenario is divided into six scenarios to see the effect of applying stemming, different word embedding vectors, and using NLP's state-of-the-art Google's Transformer. Transformer implementation in this research is made by using the Simpletranformers Python library. The Simpletransformers is set using configuration as follows:

Table 5. Deep Learning Scenario Configurations

Configuration Name	Parameters
Configuration 1	using stopwords removal, stemming, and Fasttext word embedding
Configuration 2	using stopwords removal, without stemming, and Fasttext embedding
Configuration 3	using stopwords removal, stemming, and Word2Vec embedding
Configuration 4	using stopwords removal, without stemming, and Word2Vec embedding
Configuration 5	using stopwords removal and Transformer
Configuration 6	using stopwords removal, stemming, and Transformer

The architecture used in this experiment consists of an input layer that has 128 dimensions, then followed by an embedding layer formed by word embedding (built based on Word2Vec vectors from the dataset and using 300 dimensions). Right after the embedding layer, the model is followed by a stack of SimpleRNN Keras layers for the RNN model and an LSTM standard layer for the LSTM model. The GRU layer is also a standard GRU Keras layer, while for the BiLSTM layer, a Bidirectional layer of LSTM with return_sequences is used. The stacked model was then followed by some Dense Keras layer using the relu activation function. The last Dense layer is for the classification decision maker using the sigmoid activation function because it is categorized as a binary classification problem ('spam' and 'not spam' class). All the details hyperparameters configuration of all DL layers can be seen in Table 6, while the results of accuracy, precision, recall, and F1 scores are written in Tables 7 and 8.

Table 6. The Hyperparameters in The Experiments

DL Layer	Variable	Values	DL Layer	Variable	Values
Embedding layer	min count	1	Model	optimizer	adam
	size (dimension)	300		loss	binary cross-entropy
	Iteration	100	Metrics evaluation	metrics	accuracy, precision, recall, the area under the curve, and F-1
	max features	10000		early stopping	val_loss (minimal)
	training validation	80% / 20% (11600 data / 2900 data)		epoch	50
	input length	128		batch size	32
	input dimension	128	Computer/software specs	processor	Core I5

SimpleRNN, LSTM, GRU layer	return sequences	true	RAM	16 GB
Dense layer	activation	relu and sigmoid	Tensorflow	2.3
	input	length of training (11600)	GPU	NVIDIA 2 GB
	input	30 %		

Table 7. Result of Performance Evaluation using Deep Learning Method (Configuration 1)

DL Method	Acc	Loss	Prec	Recall	AUC	F1 Score
RNN	0.63	0.65	0.63	0.99	0.5	0.77
LSTM	0.91	0.31	0.95	0.9	0.94	0.93
BI-LSTM	0.9	0.3	0.92	0.92	0.95	0.92
GRU	**0.91**	**0.4**	**0.96**	**0.89**	**0.94**	**0.92**

Table 8. Result of Performance Evaluation using Deep Learning Method (Configuration 2)

DL Method	Acc	Loss	Prec	Recall	AUC	F1 Score
RNN	0.63	0.63	0.63	1	0.5	0.7
LSTM	0.89	0.3	0.94	0.89	0.94	0.91
BI-LSTM	**0.9**	**0.4**	**0.93**	**0.91**	**0.94**	**0.92**
GRU	0.89	0.38	0.91	0.91	0.93	0.91

Table 9. Result of Performance Evaluation using Deep Learning Method (Configuration 3)

DL Method	Acc	Loss	Prec	Recall	AUC	F1 Score
RNN	0.9	0.4	0.93	0.9	0.94	0.92
LSTM	**0.9**	**0.4**	**0.93**	**0.92**	**0.94**	**0.93**
BI-LSTM	**0.9**	**0.4**	**0.93**	**0.92**	**0.94**	**0.92**
GRU	0.9	0.4	0.93	0.91	0.94	0.92

Table 10. Result of Performance Evaluation using Deep Learning Method (Configuration 4)

DL Method	Acc	Loss	Prec	Recall	AUC	F1 Score
RNN	0.9	0.3	0.95	0.89	0.94	0.92
LSTM	0.9	0.4	0.95	0.89	0.94	0.92
BI-LSTM	**0.9**	**0.4**	**0.93**	**0.91**	**0.95**	**0.92**
GRU	**0.9**	**0.4**	**0.93**	**0.91**	**0.95**	**0.92**

Table 11. Result of Performance Evaluation using Deep Learning Method in Configuration 5 and 6

Transformer DL Variant	Acc	Loss	Prec	Recall	AUC	F1 Score
SimpleTrans Bert : cahya/bert-base-indonesian-522M (Configuration 5)	**0.94**	**0.15**	**0.97**	**0.93**	**0.94**	**0.96**
SimpleTrans Roberta: cahya/roberta-base-indonesian-522M (configuration 5)	0.93	0.17	0.96	0.92	0.92	0.95
SimpleTrans Bert: cahya/bert-base-indonesian-522M (configuration 6)	**0.94**	**0.16**	**0.97**	**0.96**	**0.94**	**0.96**
SimpleTrans Roberta: cahya/roberta-base-indonesian-522M (configuration 6)	0.93	0.17	0.92	0.96	0.95	0.94

Table 6-11 shows that implementation of ML and DL algorithms can achieve well with reasonably good accuracy results. All are above 88%, except SimpleRNN. ML methods achieve an accuracy of 0.93 by Linear SVM, Linear Regression, SGD, and Extreme Tree. Bi-LSTM and GRU only achieved an accuracy of 0.9 and an F1 score of 0.92, but The Transformer (Simpletransformers) method outperformed the others. The results of ML and DL are not significantly different, but deep learning methods are still better in all the performance: accuracy, precision, recall, and F1 score. The best deep learning algorithm is obtained by Transformers (RoBERTa-based) with an accuracy of 0.94. Based on Table 9, using configuration 6 (stopwords and stemming), the accuracy is still the same, but the recall is up and reaches 0.9.

The scenario using Fasttext in the embedding layer results in low accuracy, only 63%. This result needs to be investigated further; however, the system can classify the spam class better than the non-spam class with higher accuracy, recall, and F1 score. The difference between balanced and unbalanced datasets in accuracy is just 0.05. The time elapsed for ML and DL training varies from 5 minutes to 8 hours. The DL methods are prolonged in training time because of the computational complexity, but the ML is fast. ML methods are pretty old compared to their peers, the Extra Tree and eXtreme Gradient Boosting methods. While the MLP method, although included in the ML method, is a basic DL, so the process also takes a very long time compared to other ML methods. The Transformers model has the longest training time and evaluation, but the performance is the best. ML algorithm has an average training time (acceptable) with good results (on average, 0,86).

The limitation of this study is that all experiments only used the comment text and did not use emojis. In this experiment, comment text is only treated as stand-alone data and is not related to the posting data. The post text has not been used to view the context of comments on a particular post. In future works, emojis will still be explored, and the combined use of post and comment text as a single data unit will be carried out. A comment is called spam (irrelevant) to post data if the detection process is carried out in the post's context. The spam detection process will be treated as a sub-task classification called sentence-pair classification in further research to get the context.

4. Conclusion

This research has analyzed the importance of detecting spam content on social media, mainly focusing on social media Instagram in its comments. The spam comment in question is when the comment is not related/related to the post status. This research experimented with applying ML dan DL methods to detect spam comments using the IG 2017 dataset. The accuracy from ML and DL is still in the range of 0.89 – 0.94. The best machine learning methods are Linear SVM, Extra Tree, Regression, and SGD, which have an accuracy of 0.93, while deep learning architectures have the highest accuracy of 0.94 using SimpleTransformer BERT (cahya/bert-base-Indonesian-522M). The limitation of this study is that all experiments only used the comment text and did not use emojis. In this experiment, comment text is only treated as stand-alone data and is not related to the posting data. The post text has not been used to view the context of comments on a particular post. Future works will be done by developing deep learning architecture for spam comment detection using sentence-pair classification between post and comment and emoji feature, which has been rarely used in the detection/classification of text on social media.

References

[1] Databooks, “Ini Media Sosial Paling Populer Sepanjang April 2020,” *Databooks*, 2020. https://databoks.katadata.co.id/datapublish/2020/05/25/ini-media-sosial-paling-populer-sepanjang-april-2020 (accessed Nov. 04, 2020).

[2] S. Aiyar and N. P. Shetty, “N-Gram Assisted Youtube Spam Comment Detection,” *Procedia Computer Science.*, vol. 132, pp. 174–182, 2018, doi: 10.1016/j.procs.2018.05.181.

[3] A. R. Chrismanto, A. K. Sari, and Y. Suyanto, “CRITICAL EVALUATION ON SPAM CONTENT DETECTION IN SOCIAL MEDIA,” *Journal of Theoretical and Applied Information Technology (JATIT),* vol. 100, no. 8, pp. 2642–2667, 2022, [Online]. Available: http://www.jatit.org/volumes/Vol100No8/29Vol100No8.pdf

[4] A. Chrismanto and Y. Lukito, “Klasifikasi Komentar Spam Pada Instagram Berbahasa Indonesia Menggunakan K-NN,” in *Seminar Nasional Teknologi Informasi Kesehatan*

(SNATIK), 2017, pp. 298–306.
[5] F. Prabowo and A. Purwarianti, “Instagram online shop’s comment classification using statistical approach,” in *Proceedings - 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering, ICITISEE 2017*, 2018, pp. 282–287. doi: 10.1109/ICITISEE.2017.8285512.
[6] A. Chrismanto and Y. Lukito, “Deteksi Komentar Spam Bahasa Indonesia Pada Instagram Menggunakan Naive Bayes,” *Jurnal Ultima*, vol. 9, no. 1, pp. 50–58, 2017, doi: 10.31937/ti.v9i1.564.
[7] W. Zhang and H.-M. Sun, “Instagram Spam Detection,” in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, Jan. 2017, pp. 227–228. doi: 10.1109/PRDC.2017.43.
[8] B. Priyoko and A. Yaqin, “Implementation of naive bayes algorithm for spam comments classification on Instagram,” in *2019 International Conference on Information and Communications Technology, ICOIACT 2019*, 2019, pp. 508–513. doi: 10.1109/ICOIACT46704.2019.8938575.
[9] N. A. Haqimi, N. Rokhman, and S. Priyanta, “Detection Of Spam Comments On Instagram Using Complementary Naïve Bayes,” *IJCCS (Indonesian Journal of Computing and Cybernetics Systems*, vol. 13, no. 3, p. 263, Jul. 2019, doi: 10.22146/ijccs.47046.
[10] A. Chrismanto and Y. Lukito, “Identifikasi Komentar Spam Pada Instagram,” *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, vol. 8, no. 3, p. 219, 2017, doi: 10.24843/lkjiti.2017.v08.i03.p08.
[11] A. Chrismanto, Y. Lukito, and A. Susilo, “Implementasi Distance Weighted K-Nearest Neighbor Untuk Klasifikasi Spam dan Non-Spam Pada Komentar Instagram,” *Jurnal Edukasi dan Penelitan Informatika*, vol. 6, no. 2, p. 236, 2020, doi: 10.26418/jp.v6i2.39996.
[12] A. Chrismanto, W. Raharjo, and Y. Lukito, “Design and Development of REST-Based Instagram Spam Detector for Indonesian Language,” *Proceedings - 2018 International Seminar on Application for Technology of Information and Communication: Creative Technology for Human Life*, iSemantic 2018*, iSemantic 2018*, pp. 345–350, Sep. 2018, doi: 10.1109/ISEMANTIC.2018.8549725.
[13] A. R. Chrismanto, W. Sudiarto, and Y. Lukito, “Integration of REST-Based Web Service and Browser Extension for Instagram Spam Detection,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 12, 2018, doi: 10.14569/IJACSA.2018.091253.
[14] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis, “An up-to-date comparison of state-of-the-art classification algorithms,” *Expert Systems with Applications.*, vol. 82, pp. 128–150, 2017, doi: 10.1016/j.eswa.2017.04.003.
[15] M. P. Nugraha, A. Nurhadiyatna, and D. M. S. Arsa, “Offline Signature Identification Using Deep Learning and Euclidean Distance,” *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, vol. 12, no. 2, pp. 102–111, Aug. 2021, doi: 10.24843/LKJITI.2021.V12.I02.P04.
[16] I. P. A. E. D. Udayana, M. Sudarma, and N. W. S. Ariyani, “Detecting Excessive Daytime Sleepiness With CNN And Commercial Grade EEG,” *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, vol. 12, no. 3, pp. 186–195, Nov. 2021, doi: 10.24843/LKJITI.2021.V12.I03.P06.
[17] P. K. Roy, J. P. Singh, and S. Banerjee, “Deep learning to filter SMS Spam,” *Future Generation Computer Systems*, vol. 102, pp. 524–533, 2020, doi: 10.1016/j.future.2019.09.001.
[18] S. Dutta, T. Saha, S. Banerjee, and S. K. Naskar, “Text normalization in code-mixed social media text,” *2015 IEEE 2nd International Conference on Recent Trends in Information Systems,* ReTIS 2015 - Proceedings, no. c, pp. 378–382, 2015, doi: 10.1109/ReTIS.2015.7232908.
[19] A. Chandra and S. K. Khatri, “Spam SMS Filtering using Recurrent Neural Network and Long Short Term Memory,” *2019 4th International Conference on Information Systems and Computer Networks*, ISCON 2019*, ISCON 2019*, pp. 118–122, 2019, doi: 10.1109/ISCON47742.2019.9036269.
[20] T. Wu, S. Wen, Y. Xiang, and W. Zhou, “Twitter spam detection: Survey of new approaches and comparative study,” *Computers & Security*, vol. 76, pp. 265–284, Jul. 2018, doi: 10.1016/j.cose.2017.11.013.
[21] A. A. Septiandri and O. Wibisono, “Detecting spam comments on Indonesia’s Instagram

posts," *Journal of Physics: Conference Series*, vol. 801, no. 012069, pp. 1–7, 2017, doi: 10.1088/1742-6596/755/1/011001.

[22] R. Wongso, F. A. Luwinda, B. C. Trisnajaya, O. Rusli, and Rudy, "News Article Text Classification in Indonesian Language," *Procedia Computer Science*, vol. 116, pp. 137–143, 2017, doi: 10.1016/j.procs.2017.10.039.

[23] F. Z. Ruskanda, "Study on the Effect of Preprocessing Methods for Spam Email Detection," *Indonesian Journal on Computing (Indo-JC)*, vol. 4, no. 1, p. 109, 2019, doi: 10.21108/indojc.2019.4.1.284.

[24] W. Etaiwi and G. Naymat, "The Impact of applying Different Preprocessing Steps on Review Spam Detection," *Procedia Computer Science*, vol. 113, pp. 273–279, 2017, doi: 10.1016/j.procs.2017.08.368.

[25] C. Mus, "10+ Akun Instagram Dengan Followers Terbanyak Di Indonesia," *musdeoranje.net*, 2015. http://www.musdeoranje.net/2016/08/akun-instagram-dengan-followers-terbanyak-di-indonesia.html (accessed Oct. 13, 2021).

[26] D. Mekala and J. Shang, "Contextualized Weak Supervision for Text Classification," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 323–333. doi: 10.18653/v1/2020.acl-main.30.

[27] K. Hammar, S. Jaradat, N. Dokoohaki, and M. Matskin, "Deep Text Mining of Instagram Data without Strong Supervision," *Deep Text Mining of Instagram Data without Strong Supervision*, pp. 158–165, 2019, doi: 10.1109/WI.2018.00-94.

[28] H. Zhang, "The Optimality of Naive Bayes," in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, 2004, pp. 562–567. [Online]. Available: http://www.aaai.org/Library/FLAIRS/2004/flairs04-097.php

[29] Scikit-Learn, "1.4. Support Vector Machines — scikit-learn 0.23.2 documentation," *Scikit-Learn Documentation*, 2021. https://scikit-learn.org/stable/modules/svm.html (accessed Nov. 19, 2020).

[30] Suyanto;, *Data mining untuk klasifikasi dan klasterisasi data*, 1st ed. Bandung: Informatika, 2017. Accessed: Nov. 19, 2020. [Online]. Available: //catalogue.ubharajaya.ac.id/slims/index.php?p=show_detail&id=39879

[31] J. Han, M. Kamber, and J. Pei, *Data Mining : Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011. Accessed: Nov. 19, 2020. [Online]. Available: https://www.amazon.com/Data-Mining-Concepts-Techniques-Management/dp/0123814790

[32] P. Soucy and G. W. Mineau, "A simple KNN algorithm for text categorization," *Proceedings - IEEE International Conference on Data Mining*, ICDM*, ICDM*, pp. 647–648, 2001, doi: 10.1109/icdm.2001.989592.

[33] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997, doi: 10.1006/jcss.1997.1504.

[34] N. Bhandari, "A Gentle Introduction to XGBoost for Applied Machine Learning," *Medium*, 2018. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (accessed Dec. 16, 2020).

[35] J. Brownlee, "ExtraTreesClassifier. How does ExtraTreesClassifier reduce… | by Naman Bhandari | Medium," *Machine Learning Mastery*, 2016. https://medium.com/@namanbhandari/extratreesclassifier-8e7fc0502c7 (accessed Dec. 16, 2020).

[36] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach Learn*, vol. 63, pp. 3–42, 2006, doi: 10.1007/s10994-006-6226-1.

[37] R. N. Waykole and A. D. Thakare, "A Review of Feature Extraction Methods for Text Classification," *International Journal of Advance Engineering and Research Development*, vol. 5, no. 04, pp. 351–354, 2018.

[38] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, pp. 3483–3487, 2019.

[39] P. Liu, X. Qiu, and H. Xuanjing, "Recurrent neural network for text classification with multi-task learning," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2016-Janua, pp. 2873–2879, 2016.

[40] Y. Lukito and A. Chrismanto, "Recurrent neural networks model for WiFi-based indoor positioning system," in *2017 International Conference on Smart Cities, Automation &*

Intelligent Computing Systems (ICON-SONICS), Nov. 2017, vol. 2018-Janua, pp. 121–125. doi: 10.1109/ICON-SONICS.2017.8267833.

[41] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[42] A. W. Ramadhan, D. Adytia, D. Saepudin, S. Husrin, and A. Adiwijaya, “Forecasting of Sea Level Time Series using RNN and LSTM Case Study in Sunda Strait,” *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, vol. 12, no. 3, p. 130, 2021, doi: 10.24843/lkjiti.2021.v12.i03.p01.

[43] K. Cho *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1724–1734, 2014, doi: 10.3115/v1/d14-1179.

[44] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transaction Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997, doi: 10.1109/78.650093.

[45] A. Vaswani *et al.*, “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.

[46] D. Bahdanau, K. H. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.