ISSN 2085-4552

Spam Filtering on User Feedback via Text Classification using Multinomial Naïve Bayes and TF-IDF

Septiyan Mudhiya Sadid[1], Julio Christian Young[2], Andre Rusli[3]
Department of Informatics, Multimedia Nusantara University, Tangerang, Indonesia
[1]septiyan.mudhiya@student.umn.ac.id, [2]julio.christian@umn.ac.id, [3]andre.rusli@lecturer.umn.ac.id

Accepted 18 June 2021
Approved 12 September 2021

***Abstract*— User feedback could give a developer information on what should be fixed or should be improved. But there are many users feedback that is a spam. In user feedback, spam contents are more likely to be inappropriate feedback, feedback that is not feedback, just some random comment or even a question. Reading and choosing feedback manually could be costly, especially in terms of time and energy. Therefore, this research focuses on building a spam filtering model using Multinomial Naïve Bayes that implement a TF/IDF approach to detect spam automatically. For text classification, Multinomial Naïve Bayes proved on having better speed and having good performance. With TF/IDF, a word that highly occurred in many documents has less impact than others so it could help increasing performance from an imbalanced dataset. This research aims to implement Multinomial Naïve Bayes for spam filtering in user feedback and to measure the performance of the model. The best performance of this classifier was obtained when using the up-sampling method and typo corrector with a 70:30 ratio of train and test set resulting in 89.25% for accuracy, 45% for precision, 56% for recall, and 50% for F1-Score.**

Index Terms— Spam filtering, Multinomial Naïve Bayes, User Feedback, TF/IDF, Requirements Engineering

I. INTRODUCTION

In software development, user requirements is a useful component that developers needed. In order to correctly analyse and define the users' problem and needs, requirements engineering has become an important step in a software development lifecycle. Requirements engineering is a branch of software engineering which deals with problems that involve goals, functions, and restrictions on software systems [1]. Requirements engineering help software developer to have a better understanding about the problems. One of many ways to do requirements engineering is to directly involve the users in the development process. In the case in which the users of a software product is the public, it would be difficult to get all the users to be involved directly in the requirements specification and definition process. Many software developers rely on the users feedback gathered from many different sources online. For example, a mobile app developer in the Android environment would gather a lot of user feedback from the Google Play Store, and iOS environment from the Apple App Store. Another example is the web application of an e-commerce platform, the developers usually provide comment sections in order to gather feedbacks from their users, albeit content-related, product-related, or other categories of feedback.

User feedback is defined as all information obtained from users about whether they are satisfied or not with a product or service [2]. Users will give feedback if they are satisfied or there is a problem when they use the software. Even though user feedback has a lot of benefits for software development, some problems could occur such as spam in user feedback. Spam is an activity to send a message to other people with an electronic device continuously without the consent of the other party [3]. Spam is usually targeting random people. Spam messages always have the same characteristics, therefore it is not too hard to distinguish. This is particularly a serious problem for software products with thousands or even millions of user feedback. Manually reading each of the feedback could take time and effort which otherwise could be used for other development activities. It is more time-wasting if the feedback themselves are actually spam messages.

In user feedback, spam is more like inappropriate feedback, which is the feedback that is not actual feedback such as insulting feedback, fake review, etc. Inappropriate feedback is categorized as spam because that feedback is not useful for requirements engineering. Therefore, a classification system needs to be made to predict which feedback is spam. This

research will use TF-IDF along with Multinomial Naive Bayes to make a classification system to detect spam in user feedback.

II. Related Work

Multinomial Naive Bayes is an improvement from Naive Bayes Classifier. Naive Bayes itself is a classification method that uses Bayes Theorem where each feature is assumed as independent to each other [4]. According to research from [5], the Multinomial Naïve Bayes algorithm runs 2 to 6 times faster than the Support Vector Machine. Multinomial Naive Bayes do better performance in 9 out of 13 models that [5] have. According to research from [6], the Multinomial Naïve Bayes algorithm has 89.58% accuracy in the letter classification system.

When dealing with text classification, it is necessary to do feature extraction before the classification. One of many popular feature extractions on text classification is TF-IDF and Doc2Vec. According to [7], TF-IDF has a better performance than Doc2Vec with 95% accuracy on Logistic Regression and 73.62% accuracy on Naïve Bayes. From the previous works, experiments related to the use of Multinomial Naïve Bayes and TF-IDF, which have been proven to be effective in other cases, for filtering spams in user feedback is still rare to be found. The main contributions of this research is to implement TF-IDF along with Multinomial Naïve Bayes to classify user feedback in Bahasa Indonesia and measure the performance of the system. Based on the results, we hope to be able to give more options to the developers to better their requirements engineering activities, especially when dealing with user feedback. Hopefully, by automating the spam filtering activity, more of the developers time and effort could be spent on other productive and creative activities to improve their products.

III. Research Methodologies

A. Dataset

Dataset used in this research is user feedback from tiket.com app in Bahasa. Dataset is gathered using an automatic scrapping extension on the google play store. Then, the dataset is labelled by 5 of our colleagues with a majority vote. The total dataset used in this research is 900 data with 810 data of Ham and 90 data of Spam. The distribution of the dataset can be seen in Fig. 1.

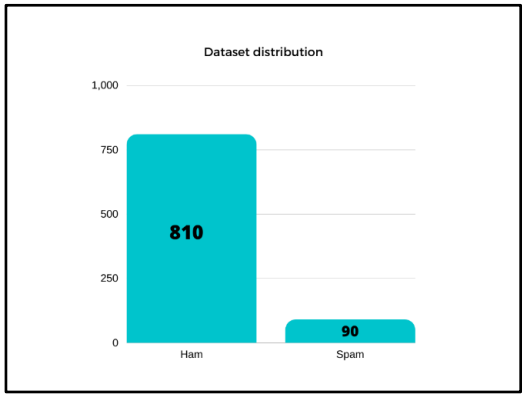

Fig. 1. Dataset distribution

B. Preprocessing

Preprocessing involved in this research is consist of Tokenizing, Filtering, and Stemming.

Tokenizing separates each sentence into an array of words. For example, the sentence “There are 2 winners of this game” will be tokenized like (‘There’, ’are’, ’2’, ’winners’, ’of’, ’this’, ’game’).

Filtering is a process that removes a bunch of words that is not useful in this research such as I, you, then. With stemming, words are reduced to their word stems [8]. For example, the sentence “He was swimming in the pool” will be reduced to “He is swim in the pool”.

C. TF-IDF

TF-IDF is an algorithm that is based on statistical values showing the appearance of a word in the document [9]. TF or Term-Frequency states how many words appear in a document. Meanwhile, IDF or Inverse Document Frequency states the number of documents that contain a word in one publication segment [6].

While computing TF, all words are treated equally important. However, it is known that certain words, such as “is”, “the”, and “and”, may appear a lot of times but have little importance. Therefore, we need to decrease the frequent terms while increasing the rare ones, by computing IDF, an inverse document frequency factor is integrated which depreciate the weight of words that occur very frequently in the document set and scale up the weight of words that occur rarely.

IDF is the inverse of the document frequency which measures the informativeness of word t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as “is” is present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

Now there are few other problems with the IDF, in the case of a large corpus, the IDF value will explode. To avoid this effect, we use the log of the IDF. When a word that is not in vocab occurs, the DF will be 0. To avoid that, we add +1 to the denominator. Here is the final formula:

$$TF - IDF(w,d) = TF(w,d) \times IDF(w) \quad (1)$$

$$IDF(w) = log\ (\frac{N}{DF(w)}) \quad (2)$$

Where:

- *TF - IDF (w, d)*: weight of a word in all documents.
- *w*: a word
- *d*: a document
- *TF (w,d)*: the frequency of the occurring word *w* in document *d*
- *IDF (w)*: inverse *DF* from word *w*
- *N*: a total of document
- *DF (w)*: total document that has word *w*

D. Naive Bayes Classifier

Naive Bayes Classifier is a classification algorithm based on the Bayes theorem. Naive Bayes Classifier assumes that every word in a document is independent. That is the presence of one particular feature does not affect the other [4]. Bayes theorem formula is shown in (3).

$$P(B) = \frac{P(A)x\ P(A)}{P(B)} \quad (3)$$

E. Multinomial Naive Bayes

Multinomial Naive Bayes is very similar to Naive Bayes Classifier. Multinomial Naive Bayes calculate the multinomial distribution from each feature in the documents [10]. The probability of a document *d* being in class *c* is computed as:

$$P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c) \quad (4)$$

Where P(tk|c) is the conditional probability of term tk occurring in a document of class c. We interpret P(tk | c) as a measure of how much evidence tk contributes that c is the correct class. P(c) is the prior probability of a document occurring in class c. If a document's words do not provide clear evidence for one class versus another, we choose the one that has a higher prior probability. The formula of Multinomial Naïve Bayes is shown in (5).

$$Cmap = arg\ arg\ P(c) \prod_{k=1}^{m} P(t_k\ |c) \quad (5)$$

Parameter P(tk | c) (probability likelihood) is estimated by calculating the occurrence of tk on all training documents in c, using laplacean prior as shown in (6) [6]:

$$P(t_k \mid c) = \frac{1+ N_k}{|V|+N} \quad (6)$$

Where Nk counts the total occurrence of tk in c's training document and N is the total occurrence of words in c [6]. For example, given training this training data:

TABLE I. TRAIN DATA

Text	Class
Free money, click the link now	Spam
Where are you now?	Ham
I am busy now, call later	Ham
Click the link and get free souvenir from me	Spam

Then we want to classify the sentence "Call me for free item" where the target class is spam or ham. First, we will calculate the probability likelihood of every word in the sentence using (6).

TABLE II. PROBABILITY LIKELIHOOD

Word	P (word | spam)	P(word | ham)
Call	$\frac{0+1}{14+18}$	$\frac{1+1}{14+18}$
Me	$\frac{1+1}{14+18}$	$\frac{0+1}{14+18}$
For	$\frac{0+1}{14+18}$	$\frac{0+1}{14+18}$
Free	$\frac{2+1}{14+18}$	$\frac{0+1}{14+18}$
Item	$\frac{0+1}{14+18}$	$\frac{0+1}{14+18}$

Based on the calculation in Table 1, we can calculate the posterior probability of the sentence.

$$P(Call|spam) \times P(Me|spam) \times P(For|spam) \times P(Free|spam) \times P(Item|spam) = \frac{1}{32} \times \frac{2}{32} \times \frac{1}{32} \times \frac{3}{32} \times \frac{1}{32} = 0.000000178813934$$

$$P(Call|ham) \times P(Me|ham) \times P(For|ham) \times P(Free|ham) \times P(Item|ham) = \frac{2}{32} \times \frac{1}{32} \times \frac{1}{32} \times \frac{1}{32} \times \frac{1}{32} = 0.0000000596046448$$

Based on the result above, we can conclude that the sentence "Call me for free item" is classified as Spam because the sentence has a higher posterior probability in class Spam.

F. The Learning Process

This research uses TF-IDF along with Multinomial Naïve Bayes to classify a text as spam or ham. The learning process of the system can be seen in the following section.

1. First, we load the file that contains the list of text that want to be classified.
2. Then we begin the pre-processing that include tokenization, filtering, and stemming.
3. We also use stop-words to remove the list of words that occur too often.
4. When the pre-processing is over, we divide the dataset to train and test dataset.
5. And then we made a vocabulary of features as a training process.
6. After that, we convert the training dataset and test dataset to the TF-IDF model.
7. Finally, we fit the model to the Multinomial Naïve Bayes algorithm and then calculate the performance of the model.

G. Confusion Matrix

A confusion matrix is a method to measure the performance of the classifier. The confusion matrix is in the form of a table with 4 different combinations of predicted and actual values [11].

TABLE III. CONFUSION MATRIX

Predicted \ Actual	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Where TP occurred when the system predicted positive, and the actual data is positive (True Positive). TN means True Negative, where the system predicted negative, and the actual data is negative. FP occurred when the system predicted positive, but the actual data is negative (False Positive). FN means False Negative, where the system predicted negative, but the actual data is positive [11].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (7)$$

$$Precision = \frac{TP}{(TP+FP)} \quad (8)$$

$$Recall = \frac{TP}{(TP+FN)} \quad (9)$$

$$F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision+Recall} \quad (10)$$

Confusion matrix can be used to calculate precision, recall, F1-Score, and accuracy of the classifier. Accuracy is used to measure the accuracy of predictions made by the classifier. Precision calculates the ratio of true positive predictions to all positive predictions. While recall calculates the ratio of true positive predictions to all predictions in the actual class.

IV. RESULT AND DISCUSSION

A. Performance Evaluation

Several testing scenarios are used in this research to determine the best approach to classify spam feedbacks. The first scenario is to compare the performance of different ratios of train and test sets (Scenario 1). The next scenario is using upsampling and downsampling train dataset (Scenario 2). Upsampling is done by increasing the minority class data to the same amount as the majority class. Downsampling is done by reducing the majority class data to the same amount of minority classes. And then we compare both methods to determine what is the best method for our research. The last scenario is we use a typo-corrector from [12] in our research (Scenario 3). A typo-corrector is a method to fix a typo in the text. Our dataset has many words that contain a typo. Therefore, with the use of a typo-corrector, we believe it will increase the performance of the model.

In Scenario 1, we compare the performance from the 70:30 ratio of train and test set with 80:20. According to the result of both methods, each method obtained low precision, recall, and F1-Score. That is because the dataset used in this research is imbalanced. However, the best performance in this scenario is the ratio of 70:30 with 50% precision, 3% recall, and 6% F1-Score on spam class. The performance of this scenario can be seen in the table below.

TABLE IV. PERFORMANCE OF SCENARIO 1

Metrics	70:30		80:20	
	Spam	Ham	Spam	Ham
Precision	50%	89%	0%	91%
Recall	3%	100%	0%	99%
F1-Score	6%	94%	0%	95%

The best performance of this scenario has an accuracy of 94.3% in the training dataset and 92.59%

in the test dataset. The overall accuracy of this scenario can be seen in the table below.

TABLE V. ACCURACY OF SCENARIO 1

Metrics	70:30		80:20	
	Train	Test	Train	Test
Accuracy	94.3%	92.59%	90%	90%

Because our dataset is imbalanced, then our next scenario is applying an up-sampling and down-sampling method to our train data. With up-sampling and down-sampling, the performance of this imbalanced dataset is better than in the previous scenario. Up-sampling method got 46% precision, 44% recall, and 45% F1-Score. The down-sampling method got 25% precision, 67% recall, and 37% F1-Score on spam class. Up-sampling method has higher performance in precision and F1-Score, while the down-sampling method has higher scores in the recall. Therefore, it is necessary to decide which metric is a better value in this case.

In requirements engineering, user feedback is used by developers to find out what user needs of the application are used. Therefore, in this case, recall is a better metric in our research. However, when dealing with an imbalance dataset we can not only look at the value of precision or recall. Because our dataset is imbalanced, the recall value could not be considered accurate. We need to consider another value that is a harmonic value of both precision and recall, which is F1-Score. According to the F1-Score value and other overall scores, up-sampling is a better method in our research.

The performance and accuracy of scenario 2 can be seen in the table below.

TABLE VI. PERFORMANCE OF SCENARIO 2

Metrics	Up-sampling		Down-sampling	
	Spam	Ham	Spam	Ham
Precision	46%	94%	25%	94%
Recall	44%	93%	67%	78%
F1-Score	45%	94%	37%	81%

TABLE VII. ACCURACY OF SCENARIO 2

Metrics	Up-sampling		Down-sampling	
	Train	Test	Train	Test
Accuracy	98,67 %	88,88 %	98,41 %	77,03 %

In scenario 3, we apply typo-corrector to our dataset. In our dataset, there are many words used by users that are not listed in the dictionary and there are many typo errors. When we try to apply the typo-corrector to our dataset, it shows a better performance in our model. In this scenario, our model has the best performance of 89.25% accuracy, 45% precision, 56% recall, and 50% F1-Score on spam class as shown in Table 8. From this scenario, we could say that the current typo-corrector improves the performance of the model. However, the improvement is not significant. The result of this scenario could be seen in Table 8 and Table 9.

TABLE VIII. PERFORMANCE OF SCENARIO 3

Metrics	With Typo-corrector		Without Typo-corrector	
	Spam	Ham	Spam	Ham
Precision	45%	95%	45%	94%
Recall	56%	93%	48%	93%
F1-Score	50%	94%	46%	94%

TABLE IX. ACCURACY OF SCENARIO 3

Metrics	With Typo-corrector		Without Typo-corrector	
	Train	Test	Train	Test
Accuracy	98,49 %	89,25 %	98,67%	88,88 %

TABLE X. BEST MODEL PERFORMANCE

Metrics	Spam	Ham
Precision	45%	95%
Recall	56%	93%
F1-Score	50%	94%
Accuracy	89.25%	

Based on the test scenarios that have been done before, we conclude that using an upsampling method with the ratio of 70:30 in training and test data as well as the use of a typo-corrector succeeded in improving model performance that has an imbalanced dataset. The best model of these scenarios has 45% precision, 56% recall, 50% F1-Score, and 89.25% accuracy on spam class. For the ham class, this model has 95% precision, 93% recall, and 94% F1-Score.

V. CONCLUSION AND FUTURE WORKS

In this research, we used TF-IDF as feature extraction and Multinomial Naive Bayes to classify whether the user feedback of tiket.com application is spam or not. Our dataset consists of 810 ham and 90 spam data that are gathered using the scrapping method from the google play store. We use a confusion matrix to calculate the performance of the classification model. Based on previous test scenarios, the best model is reached when we apply upsampling method with the ratio of 70:30 in training and test data as well as using typo-corrector from [12]. The best performance of the model is 89.25% accuracy, 45% precision, 56% recall, and 50% F1-Score in the spam class.

In future works, we suggest adding more data to our current dataset because our dataset has very low spam data. We recommend that at least the ratio of minority and majority class is 40:60 so it will be balanced. Typo-corrector has been proven to improve our model. However, the improvement is still very low. An improvement to the typo-corrector could make a better performance in the classifier. Last, we suggest trying a different method of upsampling and downsampling, since that method is very useful when handling an imbalanced dataset. There are a couple of upsampling methods like SMOTE (Synthetic Minority Over-Sampling Technique) that could perform better than our previous upsampling method.

ACKNOWLEDGEMENT

We would like to express our special gratitude to the Software Engineering Laboratory in Universitas Multimedia Nusantara who supported us a lot in doing our research.

REFERENCES

[1] Wahono, R.S. (2006). *MENYEGARKAN KEMBALI PEMAHAMAN TENTANG REQUIREMENT ENGINEERING*. [Online] Available at: https://romisatriawahono.net/2006/04/29/menyegarkan-kembali-pemahaman-tentang-requirement-engineering/ [Accessed 1 Feb. 2020]

[2] Ceban, V. (2018). *User Feedback: What It Really Is and Why It's So Important to Any Business' Success?*. [Online] Available at: https://www.appzi.com/what-is-user-feedback-and-its-role/ [Accessed 1 Feb. 2020].

[3] Maxmonroe, n.d. *Arti Spam*. [Online] Available at: https://www.maxmanroe.com/vid/teknologi/arti-spam-adalah.html [Accessed 20 Sep. 2019]

[4] Gandhi, R. (2018). *Naive Bayes Classifier*. [Online] Available at: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c [Accessed 20 Sep. 2019]

[5] Matwin, S. & Sazonova, V. (2012). *Direct comparison between support vector machine and multinomial naive bayes algorithms for medical abstract classification*. [Online] Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3422847/ [Accessed 1 Feb. 2020]

[6] Kalokasari, D. H., Shofi, D.I.M. & Setyaningrum, A.H. (2017). IMPLEMENTASI ALGORITMA MULTINOMIAL NAIVE BAYES CLASSIFIER PADA SISTEM KLASIFIKASI SURAT KELUAR. *JURNAL TEKNIK INFORMATIKA*, vol. 10, no. 2, pp. 109-118

[7] Arora, I. (2017). *Document feature extraction and classification*. [Online] Available at: https://towardsdatascience.com/document-feature-extraction-and-classification-53f0e813d2d3 [Accessed 18 May 2020]

[8] Heidenrich, H. (2018). *Stemming? Lemmatization? What?*. [Online] Available at: https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8 [Accessed 20 Sep. 2019]

[9] Wijaya, A.P. & Santoso, H.A. (2016). Naive Bayes Classification pada Klasifikasi Dokumen. *Journal of Applied Intelligent System*, vol. 1, no.1, pp. 48-55

[10] Huang, O. (2017). *Applying Multinomial Naive Bayes to NLP Problems: A Practical Explanation.* [Online] Available at: https://medium.com/syncedreview/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation-4f5271768ebf [Accessed 20 Sep. 2019]

[11] Narkhede, S. (2018). *Understanding Confusion Matrix*. [Online] Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 [Accessed 20 Sep. 2019]

[12] Setiabudi, Reza. (2020). Implementasi Algoritma Levenshtein Distance untuk Typo Correction Bahasa Indonesia pada User Feedback Aplikasi. Bachelor Thesis thesis, Universitas Multimedia Nusantara.