Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.031270
Article

Tech Science Press

New Spam Filtering Method with Hadoop Tuning-Based MapReduce Naïve Bayes

Keungyeup Ji and Youngmi Kwon*

Department of Radio and Information Communications Engineering, Chungnam National University, Daejeon, 34134, Korea
*Corresponding Author: Youngmi Kwon. Email: ymkwon@cnu.ac.kr
Received: 13 April 2022; Accepted: 08 June 2022

Abstract: As the importance of email increases, the amount of malicious email is also increasing, so the need for malicious email filtering is growing. Since it is more economical to combine commodity hardware consisting of a medium server or PC with a virtual environment to use as a single server resource and filter malicious email using machine learning techniques, we used a Hadoop MapReduce framework and Naïve Bayes among machine learning methods for malicious email filtering. Naïve Bayes was selected because it is one of the top machine learning methods(Support Vector Machine (SVM), Naïve Bayes, K-Nearest Neighbor(KNN), and Decision Tree) in terms of execution time and accuracy. Malicious email was filtered with MapReduce programming using the Naïve Bayes technique, which is a supervised machine learning method, in a Hadoop framework with optimized performance and also with the Python program technique with the Naïve Bayes technique applied in a bare metal server environment with the Hadoop environment not applied. According to the results of a comparison of the accuracy and predictive error rates of the two methods, the Hadoop MapReduce Naïve Bayes method improved the accuracy of spam and ham email identification 1.11 times and the prediction error rate 14.13 times compared to the non-Hadoop Python Naïve Bayes method.

Keywords: Hadoop; hadoop distributed file system(HDFS); MapReduce; configuration parameter; malicious email filtering; Naïve Bayes

1 Introduction

Email plays a very important role in inter-company business and individual social life, and since it has a legal effect for business handling, the importance of blocking malicious email is growing. The amount of malicious email has increased to more than 45% of all email [1], resulting in a decrease in work efficiency and increasing personal damage due to phishing. In this paper, the Naïve Bayes technique, one of the machine learning methods, is applied to MapReduce programming based on the Hadoop framework that optimizes execution time using commodity hardware to improve the prediction of spam and ham with malicious email filtering. Among the open source methods that combine commodity hardware with a virtual environment, representative ones are Spark based on the in-memory method and

the disk method-based Hadoop. The Spark method is advantageous for businesses that require fast processing based on small datasets, and the disk-based Hadoop framework may be more suitable if the data are distributed to many places and occupy several terabytes of data [2].

In a brief comparison of Hadoop and Spark, it can be seen that a common feature is that they are frameworks for big data to process large data sets. Their differences are that whereas Hadoop is a disk-based framework that provides local operation, storage, and replication functions by node in a computer cluster consisting of multiple nodes for large datasets, Hadoop has excellent expandability so that the service can be very easily expanded, and Hadoop has a structure that supports data distribution and redundant storage between nodes using the MapReduce programming model [3], while Spark is a framework for processing datasets based on memory. Spark is evaluated to be about 100 times faster than Hadoop [2], but Hadoop is more efficient to process data distributed to individual nodes at the same time. Unlike Hadoop, Spark supports MapReduce, a software platform, in a memory-based method, and Hadoop is a structure that supports disk-based MapReduce [4].

Methods to shorten Hadoop execution time include hardware improvement, network infrastructure improvement, Hadoop configuration tuning, job scheduling (or task scheduling), and data locality algorithm [5]. In this paper, the Hadoop execution time was shortened by selecting 19 parameters that affect the execution time among more than 200 Hadoop configuration parameters and obtaining optimal information based on an experiment-driven method. Based on the optimized Hadoop MapReduce framework, Naïve Bayes classifier technology based on Bayesian theory belonging to the machine learning classification algorithm was used to improve the accuracy of prediction of malicious email, spam and ham email.

2 Background Knowledge and Related Work

2.1 Background Knowledge

Hadoop is a Java-based open source framework that can carry out distributed processing of massive data. It is a framework that carries out distributed storage of data in a Hadoop Distributed File System (HDFS) and processes data using MapReduce, a distributed processing system [5]. The main characteristics of Hadoop are that the system construction cost is low because there is no license fee and only a server cost is required since it is open source and that recovery processing is possible in the event of data failure because data replication is supported. The Hadoop structure consists of the utility, Hadoop Common, which is a library group, HDFS, which carries out distributed processing and storage of data, MapReduce, which carries out calculation processing, and Hadoop Yarn, which manages resources and job schedules. The HDFS manages data storage and data distribution to individual data nodes, and tracks data storage locations, and MapReduce is a software framework that manages applications that process large amounts of datasets and is responsible for determining the data nodes that should carry out calculation processing because it has the ability to process calculation. Hadoop Yarn is composed of a resource manager and a node manager and performs resource management and work schedule management [6,7].

Among the Hadoop improvement methods, tuning by Hadoop parameters is efficient, and parameter tuning methods can be classified into six types. Hadoop configuration parameters affect job performance in various ways. Hadoop currently has more than 200 parameters, among which the main parameter items for reducing Hadoop execution time include block size setting, number of map nodes setting, number of reducer job nodes setting, map output data compression, temporary data processing setting, memory buffer size setting, number of Map/Reduce task setting, etc. The six types of parameter tuning methods are rule-based, cost-modeling, simulation-based, experiment-based, machine learning, and adaptive tuning approaches. In this paper, the experiment-based approach was selected among the six type methods

because in the case of the machine learning approach, which is the optimal method, and the adaptive tuning approach, the parameter information obtained with the machine learning technique is sometimes inappropriate for the real environment and the parameter information is affected by Hadoop version changes causing cost problems. Therefore, the experiment-based approach can optimize and tune major Hadoop parameter information reflecting system environmental information as much as possible. That is, the adaptive tuning approach, which automatically extracts parameter information with the application, is affected by the Hadoop version, and it seems that the application has difficulties in automatically optimizing parameters by properly understanding the given system environment [8].

Among related papers that improve execution time by adjusting Hadoop parameters, the results of improving the execution time of Hadoop Mahout by adjusting the block size, the number of replications, and the memory buffer size were presented in [9,10]. MapReduce is a framework model for parallel processing and distributed processing of large-scale datasets in the Hadoop environment. Map function and Reduce function are performed independently [11], as in Expression Eq. (1). The Map function creates a new array from an existing array, and the Reduce function reduces the value array to one value.

$$\begin{aligned} map\_function(k_1, v_1) &\rightarrow list(k_2, v_2) \\ reduce\_function(k_2, list(v_2)) &\rightarrow list(v_3) \end{aligned} \tag{1}$$

Machine learning is divided into supervised learning, unsupervised learning, and reinforcement learning [12,13]. Bayesian theory belongs to classification, and classification is a kind of supervised learning. Classification is used to make category predictions for newly observed data through learning based on existing data. Naïve Bayes is a classifier based on the Bayesian rule, that indicates the relationship between the prior and posterior probabilities of two random variables, and the related expression is expressed in Eq. (2) [14].

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)} \tag{2}$$

– P(A): prior probability, probability of A (reason) that is determined before the outcome is produced.
– P(B|A): likelihood probability, probability for an outcome B to occur given that reason A has occurred.
– P(A|B): posterior probability, probability for reason A to occur given that outcome B has occurred.

2.2 Related Work

Hadoop is a distributed computing-based framework that processes massive data and has the characteristic of processing data merging and sorting using MapReduce characteristics. The reason is that the framework moves and merges distributed datasets across multiple data nodes and handles sorting based on key values [6]. The configuration parameter tuning approaches to improve Hadoop performance can be classified into six types: rule-based, cost-modeling, simulation-based, experiment-driven, machine learning, and adaptive. The rule-based approach supports the user's parameter tuning based on expert experience, guidelines, and tuning instruction. The cost-modeling approach attempts to improve performance by creating an efficient performance prediction model using an analysis cost function. The simulation-based approach builds a performance prediction model based on modular or system simulation and supports the user to perform simulations after changing into other parameter settings or cluster resources. The experiment-driven approach is the method adopted in this paper, and is minimally affected by the system environment such as Hadoop version and can perform parameter tuning to fit the real system environment and data size through experiments, but has the disadvantage of taking long time. The machine learning approach can perform parameter tuning through real system observation, but it is not easy to secure training data sets, and there is a burden to improve low accuracy. The adaptive approach is a method in which the application tunes the configuration parameters to fit changes in the system

environment. It is effective for ad-hoc applications, but may lead to inappropriate configuration and inefficient resource utilization [15].

To improve Hadoop performance, about five configuration parameters, including block size, number of reducers, the size of the replication node, and their set values were adjusted based on the experiment-driven approach [16,17].

Kim et al. [18] presented a method for performance tuning by dividing a single large-capacity server with large-scale performance and resources into multiple Virtual Machine(VM) servers and adjusting Hadoop configuration parameters as a performance improvement method using hardware. It is another method for processing massive data that can efficiently divide and use massive resources. Also, Jeon et al. [19] suggested a Hadoop performance tuning method by reducing the size of data transmitted to the network and minimizing disk I/O.

Spam filtering methods are largely divided into reputation-based filtering methods and content-based filtering methods. Representative methods of reputation-based filtering include the blacklist method and whitelist method. Representative methods of content-based filtering include machine-learning filtering methods, and representative examples of machine-learning methods include Bayesian filtering, SVM filtering, and boosting algorithm methods. The Bayesian filtering method was applied in this paper, and it is a method used to generate a final probability estimate by combining the probabilities of individual spams or hams of each word in the message [20]. In particular, Naïve Bayes is one of the unsupervised methods among the Bag of Words (BoW) model-based learning methods, and the Naïve Bayes formula is expressed in Eq. (3) [21].

$$c^* = \arg\max_c p(c|w) = arg\max_c p(c)p(w|c) = arg\max_c p(c)\prod_{n=1}^{N} p(w_n|c) \tag{3}$$

(w: each image is represented by w = $\{w_1, w_2, \ldots, w_n\}$, all patches are an image c: image category, $\prod$: mixture proportion).

Machine learning-based spam classification algorithms include SVM, Naïve Bayes, KNN and Decision Tree. Among them, SVM and Naïve Bayes are the most efficient methods, but they have problems such as low scalability and limited accuracy. Therefore, as a solution to the problems, accuracy, speed, and scalability were improved by performing it in the MapReduce framework [22]. As another image filtering method, Liu et al. [23] suggested a method of filtering image spam with three-layer classifiers (email header classifier, image header classifier, and feature classifier). In detail, spam filtering is performed by a Naive Bayesian classifier in the first layer, and by a SVM classifier in the second and third layers. In particular, email headers are analyzed in the first layer, and the high-level and low-level features of images are analyzed in the second and third layers.

3 Hadoop Parameter Control for Optimizing Hadoop Framework Tuning and Improved Accuracy of Malicious Email Filtering Based on MapReduce Naïve Bayes

This paper analyzes email data and predicts spam or ham, which are harmful traffic, using the Naive Bayes algorithm in the MapReduce program based on the Hadoop framework. To process massive data, a Hadoop environment was selected where scale-out is easy and existing commodity hardware is combined with a virtual environment to easily manage the distributed replication of data. To improve the prediction accuracy of spam filtering, the Naive Bayes algorithm with the Bayesian theorem applied was used. Vyas et al. [24] proved in experiments that the Naïve Bayes algorithm selected in this paper has the best speed and accuracy among spam filtering methods. According to the experimental contents, Naïve Bayes is 2.1 times faster than Iterative Dichotomiser 3(ID3), which is second among the main spam filtering techniques J48, Naïve Bayes, Sequential Minimal Optimization(SMO), and ID3. The accuracy of Naïve

Bayes is 97.7% of that of ID3, which is in first place, and Naïve Bayes works for both continuous and discontinuous properties in the same dataset.

As for the tuning method, the Hadoop execution time was improved by selecting 19 Hadoop configuration parameters, which are elements that greatly affect performance, among the Hadoop configuration parameters based on an experiment-driven approach [25,26]. The parameters applied in this paper are included in the three environment information setting files, hdfs_site.xml, mapred_site.xml, and yarn_site.xml. Performance improvement experiments were performed by adjusting the number of concurrent replication tasks, the memory size set in the container, the buffer size for sorting, the number of threads, and the number of compressions. And in Yarn-site.xml, performance improvement experiments were performed by adjusting the size of MapReduce's container, the node manager memory size, and the number of virtual CPU cores allocated to the container.

In order to shorten the execution time while increasing malicious email filtering accuracy with the improved Hadoop framework, this paper conducted experiments using the Naïve Bayes theorem based on MapReduce. According to the experimental results of Vyas et al. [24], the accuracy and execution time of each of clustering, Java48 (J48), Naïve Bayes, SVM, and ID3, which are techniques used to separate spam or ham email from normal email, were compared with each other and analyzed through experiments. As a result of the experiment, it was concluded that SMO and ID3 are superior to Naïve Bayes in terms of accuracy, but Naïve Bayes is the best in terms of execution time. The accuracy and execution time values of these methods are described in Tab. 1, the accuracy is expressed as a graph in Fig. 1, an execution time graph is shown in Fig. 2, and the calculation formula to obtain the accuracy [27] is expressed in Eq. (4).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100 \tag{4}$$

– True Positive (TP): Predicting the correct answer that is actually true as true (correct answer)
– False Positive (FP): Predicting the correct answer that is actually false as true (wrong answer)
– False Negative (FN): Predicting the correct answer that is actually true as false (wrong answer)
– True Negative (TN): Predicting the correct answer that is actually false as false (correct answer)

Table 1: Comparison between machine learning techniques [24]

Technique	Accuracy	Execution time	TP	TN	FP	FN
Clustering	53.915%	1.33 sec	1	0	1	0
J48	89.3617%	1.52 sec	0.84	0.955	0.045	0.16
Naïve Bayes	91.4894%	0.46 sec	0.84	1	0	0.16
SMO	93.617%	1.92 sec	0.88	1	0	0.12
ID3	93.617%	0.81 sec	0.88	1	0	0.12

4 Improvement of Execution Time Through Hadoop Configuration Parameter Optimization and Improvement of Malicious Email Filtering Accuracy with the Application of MapReduce Machine Learning

4.1 Experiment Specification

Four commodity PCs were used in the experimental environment, the name node and secondary name node were configured with one PC each, and the data nodes were configured with three PCs. The details of hardware are shown in Tab. 2.

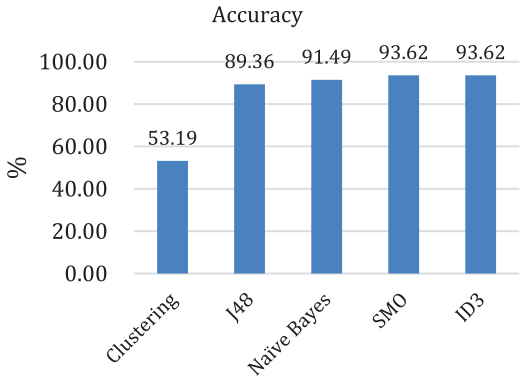

Figure 1: Accuracy comparison [24]

Figure 2: Performance comparison [24]

Table 2: Hardware specification for test

Item	Name node system spec	Data node system spec
CPU	Core i7-7700 @ 3.6 GHz	Core i5-7500 @ 3.4 GHz
Number of cores	Quad-Core	Quad-Core
OS version	Ubuntu 16.04	Ubuntu 16.04
Memory	DDR4 8 GB	DDR4 8 GB
Network environment	1 Gbps Wire LAN	1 Gbps Wire LAN
Hadoop version	2.5.2	2.5.2
Python version	2.7/3.6	3.5.2

As for the experimental method, Hadoop framework tuning was optimized by adjusting the Hadoop configuration parameters, that affect performance, and accordingly, eight test cases to be used in the test were selected. For test case 1, the default values set during Hadoop installation were used, and for test case 2 through test case 8, appropriate values for configuration parameters that affect execution time were set to perform Hadoop framework tuning. For test case 2, dfs.replication, which controls the number of dataset copies, was set to 2 and mapreduce.map.memory was set to 2048 MB and mapreduce.reduce.memory.mb was set to 4096 MB and mapreduce.map.java.opts.max.heap was set to 1638 MB and mapreduce.reduce.java.opts.max.heap was set to 3277 MB and mapreduce.task.io.sort.mb was set to 200 MB and mapred.reduce.parallel.copies was set to 20 numbers and yarn.scheduler.maximum-allocation-mb was set to 4096 MB and yarn.scheduler.maximum-allocation-vcores set to 1 and

mapreduce.output.fileoutputformat.compress.type set to block in the environment. For test case 3, unlike test case 2, dfs.block_size was changed to 128 MB and dfs.namenode.handler.count was set to 10 and dfs. replication was set to 3 and mapred.tasktracker.map.tasks.maximum was set to 7 and set mapred. tasktracker.reduce.tasks.maximum was set to 7 and mapred.map.tasks.speculative.execution was set to false. From test case 4 to test case 8, the Hadoop speed was tuned only by adjusting block size. The reason why other parameters were not adjusted is that the parameters were judged to have been optimized with the values set in test case 3. To test the execution time, dfs.block_size was set to 256 MB in test case 4, set to 512 MB in test case 5, set to 1024 MB in test case 6, set to 256 MB in test case 7 and set to 128 MB in test case 8.

As shown in Tab. 3, the execution time experiment was conducted by dividing the MapReduce program to which Laplace smoothing technology was applied and the MapReduce program to which it was not applied in the Hadoop framework environment. Laplace smoothing is a smoothing technique that helps tackle the problem of zero probability by adding 1 to the frequency of occurrence in the Naïve Bayes machine learning algorithm. The results shown in Tab. 3 were obtained when the MapReduce model was applied based on the Hadoop framework and the values of the configuration parameters were changed and tested in eight test cases. In the bare metal server environment where the Hadoop framework was not installed, the experimental data size was 158.7 MB, and the execution times by test case and the magnification of execution time of the bare metal Python application compared to that of the MapReduce application were measured using malicious email data consisting of 1.5 million cases. Referring to Surya-Murali [28] and, Mseltz [29], the programs were developed for a comparison of execution time and accuracy between the Hadoop environment and the bare metal environment to measure execution time and classify malicious email as shown in Fig. 3. In the Hadoop framework environment, the Naive Bayes classification algorithm was applied based on 1.5 million malicious email test data to implement a MapReduce program for training and a MapReduce program for classifying into spam and ham [30]. In the bare metal environment without Hadoop, a Python-based program with the Naive Bayes classification algorithm applied was implemented to measure the execution time and accuracy of classifying malicious email into spam and ham [31].

Table 3: Performance comparison between Hadoop MapReduce server and bare metal server

Test case type	With hadoop framework		Without hadoop	Execution time multiplier ratio (A/C)
	MapReduce without laplace smoothing (A) (sec)	MapReduce with laplace smoothing (B) (sec)	Bare metal server (C) (sec)	
Test case 1	5744	5771	1456	3.94
Test case 2	4980	5771	1456	3.42
Test case 3	5213	4887	1456	3.355
Test case 4	4897	5023	1456	3.362
Test case 5	5048	8352	1456	3.466
Test case 6	5032	5095	1456	3.5
Test case 7	5183	5147	1456	3.5
Test case 8	5029	5045	1456	3.5

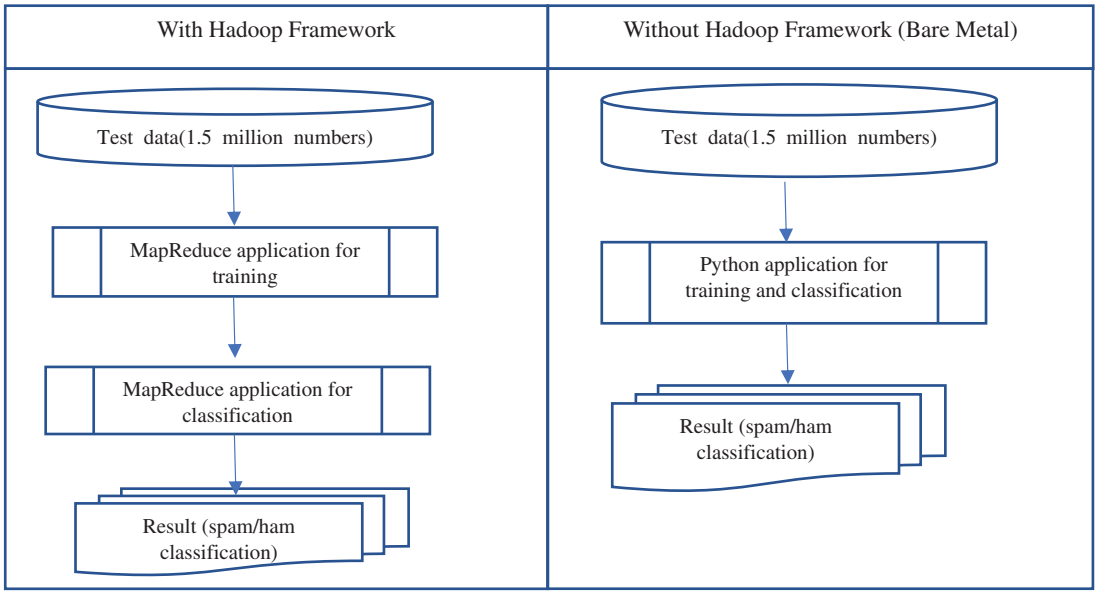

Figure 3: Test process for hadoop framework and bare metal server

Given that, the execution time in test case 1 as shown in Tab. 3 was 5744 s (1 h 36 min), which was 15 min longer compared to 4887 s (1 h 21 min) in test case 3, it can be recognized that the influence of the configuration parameter value is very large. Also, it can be recognized from the experiment that an increase in the block size parameter is not necessarily related to the improvement of execution time. For test case 6, the block size was 1024 MB, and the execution time was 5032 s, while for test case 3, the block size was 128 MB, but the execution time was 4887 s. The execution time of test case 6 was 145 s longer than that of test case 3, indicating that the performance was not improved proportionally when the block size was allocated to be larger. In addition, the experiment showed that execution time is not necessarily longer when the number of replication nodes is larger. According to the experimental results, the number of replication nodes in test case 1 was set to 1 for setting one replication, whereas in test case 2, the number of replication nodes was set to two, and from test case 3 to test case 8, the number of replication nodes was set to three, but test case 1, in which one replication node was designated, takes the longest execution time. As the number of replication nodes increases, data is duplicated and distributed to individual data nodes so that the execution time decreases thanks to the increase in data availability and data locality [32]. However, even if the number of replication nodes is continuously increased, the execution time does not change when a certain number has been reached, but rather the system load is increased due to data replications [32]. Therefore, the user may want to increase the number of replication nodes appropriately to fit the system environment. When compared with the execution time in the bare metal server environment where the Hadoop environment is not installed, it can be seen that the execution time in the bare metal server was 1456 s, which is 3431 s shorter than 4887 s for test case 3, which took the least execution time in the Hadoop environment. The reason is that the Hadoop framework operates on a virtual environment-based dataset replication and resource sharing between nodes, and distributed data processing between nodes, which seems to make the execution time in the Hadoop environment longer than the execution time in a bare metal server environment.

In Tab. 4, TP means cases where spam email is correctly predicted as spam and Hadoop MapReduce Naïve Bayes malicious email prediction program with Laplace smoothing (Hadoop MapReduce Naïve Bayes with Laplace smoothing) correctly predicted 484,977 spam emails while the bare metal Python programs correctly predicted 307,120 emails. TN means cases where ham email is correctly predicted as

ham email, and the Hadoop MapReduce Naïve Bayes with Laplace smoothing correctly predicted 998,856 ham emails and the bare metal Python Naïve Bayes malicious email prediction program in non-Hadoop environment (non-Hadoop Python Naïve Bayes) correctly predicted 999,454 ham emails. False Positive FP means cases where actual ham email is falsely predicted as spam email and the Hadoop MapReduce Naïve Bayes with Laplace smoothing falsely predicted 10,746 ham emails as spam email and the non-Hadoop Python Naïve Bayes falsely predicted 147,916 ham emails as spam email. FN means cases where actual spam email is falsely predicted as ham email and the Hadoop MapReduce Naïve Bayes with Laplace smoothing falsely predicted 10,746 spam email as ham email and the bare metal Python program falsely predicted 147,916 spam email as ham email. Based on these experimental results, the Hadoop MapReduce Naïve Bayes with Laplace smoothing made more accurate predictions than the non-Hadoop Python Naïve Bayes.

< Meaning of performance evaluation index in terms of experiment >

– TP: Correct prediction of actual spam email as spam email (correct answer)
– TN: Correct prediction of actual ham email as ham email (correct answer)
– FP: Wrong prediction of actual ham email as spam email (wrong answer)
– FN: Wrong prediction of actual spam email as ham email (wrong answer)

Table 4: Performance evaluation index comparison between both methods

Item	Hadoop MapReduce Naïve Bayes malicious email prediction with laplace smoothing	Bare Metal Python Naïve Bayes malicious email prediction in non-Hadoop
Number of file data	1,500,976	1,500,976
Number of successful file reads	1,494,579	1,454,489
Number of fail file reads	6,397	46,487
TP	484,977	307,120
TN	998,856	999,453
FP	10,746	147,916
FN	10,746	147,916
Number of correct answers	1,483,833	1,306,573
Number of incorrect answers	10,746	147,916

The performance evaluation indicators TP, TN, FP, and FN were compared graphically between the Hadoop MapReduce Naïve Bayes with Laplace smoothing and the non-Hadoop Python Naïve Bayes in Fig. 4. Based on the TP, TN, FP, and FN data shown in Tab. 4, the prediction error rate and accuracy were compared between the Hadoop MapReduce Naïve Bayes with Laplace smoothing and the non-Hadoop Python Naïve Bayes in Tab. 5. According to Tab. 5, the accuracy of Hadoop MapReduce Naïve Bayes with Laplace smoothing is higher than that of non-Hadoop Python Naïve Bayes accuracy. The accuracy calculation formula is shown in Eq. (4).

The malicious email prediction performances of Hadoop MapReduce Naïve Bayes with Laplace smoothing and non-Hadoop Python Naïve Bayes are compared in the graph shown in Fig. 5.

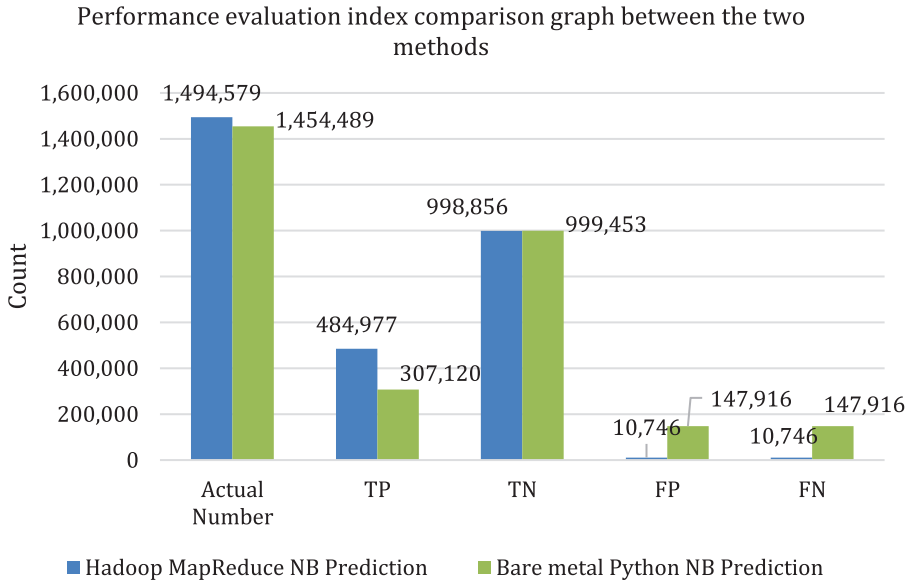

Figure 4: Performance evaluation indicator comparison graph between the two methods

Table 5: Comparison of prediction error rate and accuracy between both methods

Item	Hadoop MapReduce Naïve Bayes malicious email prediction with laplace smoothing	Bare Metal Python Naïve Bayes malicious email prediction in non-hadoop environment
Prediction error rate	0.72%	10.17%
Accuracy	99.28%	89.83%

For the algorithm for Hadoop performance improvement and malicious email prediction filtering, the MapReduce model was constructed with a two-step MapReduce method, referring to Mseltz [29]. The bare metal python algorithm was configured to perform training and malicious email prediction filtering based on the multinomial Naïve Bayes model, referring to Surya-Murali [28]. The algorithm of each program is described in Tab. 6. To prevent the probability from becoming 0 during filtering classification, 1 is added to the numerator as in Eq. (5).

$$\hat{P}(\mathrm{t}|\mathrm{c}) = \frac{T_{ct} + 1}{\left(\sum_{t' \in V} T_{ct'}\right) + B'} \tag{5}$$

– $\hat{P}(\mathrm{t}|\mathrm{c})$: the multinomial conditional probability
– $|\mathrm{V}|$: the number of terms in the vocabulary V (including all text classes)
– T_{ct}: the count of word t in class C

To classify malicious email into spam and ham, the posterior probability of class c is expressed as Eq. (6).

$$c_{map} = \arg\max_{\mathrm{c} \in \mathrm{C}} [\log \hat{P}(c) + \sum_{1 \le k \le nd} \log \hat{P}(t_k|c)] \tag{6}$$

c_{map}: Value classified as spam or ham for malicious email

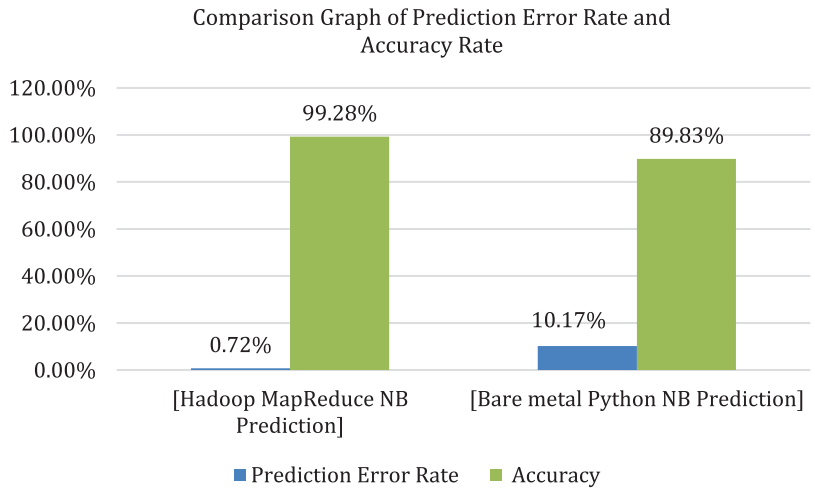

Figure 5: Comparison graph of prediction error rate and accuracy between the two methods

Table 6: Model process for malicious email prediction

Step 1) MapReduce Naïve Bayes program process with Laplace smoothing for Training
I Store test data in Hadoop directory
II Retrieve test data from Hadoop directory by Mapper program
III Remove punctuation characters (comma, period, etc.) in test data
IV Store cleaned data in Hadoop directory by Reducer program
V Training for classifying malicious data
Step 2) MapReduce Naïve Bayes program process with Laplace smoothing for classification of malicious email
I Retrieve cleaned data from Hadoop directory (from Step1 III)
II Read test data from Hadoop directory by Mapper program
III Predict whether read data is spam or ham by Reducer program
IV Classify malicious data as spam and ham
< (without Hadoop) non-Hadoop Python Naïve Bayes Malicious Email Prediction>
I Read test data from system directory
II Execute training model using test data
III Predict whether read data is spam or ham by multinomial Naïve Bayes classifier Predict (vectorize. transform(data))
IV Classify malicious data as spam and ham

The libraries used in the bare metal Python Naïve Bayes program are pandas, numpy, countvetorizer, and multinomial Naïve Bayes, and the process for filtering and classification through learning is described in Tab. 6.

4.2 Experiment Analysis

Hadoop tuning was optimized through the optimization of Hadoop configuration parameters. Execution time in the Hadoop environment was longer than that in the non-Hadoop Linux environment, because in a virtual environment, resource sharing, data replication, and data node resource allocation take longer in a virtual environment than in a bare metal environment. However, since the bare metal environment is operated by the server alone, the disadvantage is that it is difficult to share resources with other servers.

That is, there is a limit to adopting a bare metal framework for a malicious email filtering system just because the execution time is short, and a peculiarity in the Hadoop tuning process is that execution time does not necessarily improve even if the block size among configuration parameters is increased. Also, execution time does not necessarily improve just because the number of dataset copies is small. For these reasons, it is the most reasonable to derive and optimize the most appropriate configuration parameter values in a given environment for Hadoop tuning through experiments based on an experiment-driven approach.

In this paper, in addition to research to find a Hadoop performance optimization method, our study sought a method to increase the accuracy of malicious email classification by comparing and analyzing the accuracy of malicious email filtering using the MapReduce Naïve Bayes with Laplace smoothing based on the Hadoop framework optimized for tuning and the filtering system accuracy of the Python Naïve Bayes program in a non-Hadoop environment. The Hadoop MapReduce Naïve Bayes method with Laplace smoothing showed a prediction error rate that was 14.13 times smaller than the non-Hadoop Python Naïve Bayes method as a result of training and classification by applying the Laplace smoothing method. In terms of accuracy, the Hadoop MapReduce Naïve Bayes method with Laplace smoothing is 1.11 times more accurate than the non-Hadoop Python Naïve Bayes method. Therefore, for the spam and ham classification of large-capacity malicious email, the MapReduce Naïve Bayes method with Laplace smoothing in the Hadoop framework environment can be used to make more accurate predictions for malicious email filtering.

5 Conclusion and Future Work

The Hadoop framework applied in this paper has the disadvantage of being slower than the bare metal server used alone in a non-Hadoop environment because it uses several commodity servers based on a virtual environment and duplicates the dataset with a distributed computing method. In order to improve these shortcomings, eight types of test cases were configured by selecting 14 Hadoop configuration parameters that have a significant impact on performance, based on an experiment-driven approach, with various tests, and an environment with optimized performance as in test case 3 described in Tab. 2 was set. Other features of the Hadoop framework are that the efficiency of resource use is improved by combining and distributing multiple resources, the construction cost is low by using open source, there is no license restriction, and system reliability is improved by increasing data stability, as data is duplicated and stored in multiple data nodes.

For spam filtering using machine learning techniques, Naïve Bayes was chosen because it has the highest speed and accuracy among six spam filtering methods (clustering, J48, Naïve Bayes, SMO, minimal optimization, and ID3). Also, as a result of comparing the prediction error rate when a Laplace smoothing algorithm was applied to the MapReduce programming model in the Hadoop framework environment and the prediction error rate in the bare metal server environment, the Hadoop MapReduce Naïve Bayes algorithm was found to be much more accurate than the non-Hadoop Python Naïve Bayes algorithm. According to the results of analysis of eight test cases, the Hadoop MapReduce Naïve Bayes method with Laplace smoothing had a malicious email filtering's prediction error rate that was 14.13 times less than the non-Hadoop Python Naïve Bayes method and 1.11 times higher accuracy than the non-Hadoop metal Python Naïve Bayes method. As a result, it can be seen that the Hadoop MapReduce Naïve Bayes method with Laplace smoothing has higher prediction accuracy than the non-Hadoop Python Naïve Bayes method, and the Hadoop MapReduce Naïve Bayes method with Laplace smoothing is judged to be the most suitable solution for filtering malicious email.

As a future plan, we will extract sentiment information from unstructured social media test data using the Hadoop MapReduce function and machine learning methods (Naïve Bayes, SVM, and linear regression) and conduct a comparative analysis of each machine learning method through sentiment analysis.

Acknowledgement: We would like to express our very great appreciation to Department of Radio and Information Communications Engineering, Chungnam National University for providing the necessary tools and environment for this project.

Funding Statement: This work was supported by research fund of Chungnam National University.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Carey, "Hadoop and spark compared with pros and cons, ecosystem and Use cases," Itworld," 2016. [Online]. Available: http://www.itworld.co.k r/news/100370/.

[2] D. Takamori, "Apache hadoop apache software foundation documentation," Apache Software Foundation," 2006. [Online]. Available: http://hadoop.apache.org/.

[3] IBM Cloud Education, "IBM cloud education, hadoop vs. spark: What's the difference? IBM," 2021. [Online]. Available: https://www.ibm.com/cloud/blog/hadoop-vs-spark.

[4] M. Khan, "Hadoop performance modeling and job optimization for big data analytics," *Ph.D. dissertation*, Brunel University London, London, UK, 2015.

[5] B. Lee, "Binglee, concept of hadoop, DB of DB," 2018. [Online]. Available: https://dabingk.tistory.com/1?category=753768.

[6] K. Han, "Understanding Big Data and Hadoop in Do it! Hadoop with big data," 1st ed., in *Easyspublishing*, Easyspublishing, Seoul, South Korea, pp. 53–69, 2013.

[7] M. Kyu, "Mangnani-developer, structure and operation method of yarn, My study room," 2021. [Online]. Available: https://mangkyu.tistory.com/127.

[8] S. Kang, "Mapper preferences and shuffle preferences for temporary data processing, data science Lab," 2020. [Online]. Available: https://sungwookkang.com/1384.

[9] S. Jeon, H. Chung and Y. Nah, "Improving machine learning performance by Hadoop Map Reduce tuning," in *Korea Software Congress 2017*, Korean Institute of Information Scientists and Engineers(KIISE), Busan, Korea, pp. 731–733, 2017.

[10] T. D. Dinh, "Hadoop performance evaluation," in *Heidelberg Institute of Computer Science Research Group Parallel and Distributed Systems*, Ruprecht-Karls University, Baden-Württemberg, Germany, 2009.

[11] C. Chu, S. Kim, Y. Lin, Y. Yu, G. Bradski *et al.,* in *Map-Reduce for Machine Learning on Multicore*, MIT Press, Cambridge, MA 02142, US, pp. 281–288, 2007.

[12] S. Chun, in *Machine Learning Study (8) Classification Introduction*, SanghyukChun Blog, Seoul, Korea, 2015. [Online]. Available: http://sanghyukchun.github.io/64/.

[13] A. Sheshasaayee and J. Lakshmi, "Machine learning approaches on Map reduce for big data analytics," in *2015 Int. Conf. on Green Computing and Internet of Things (ICGCIoT)*, Greater Noida, India, pp. 8–10, 2015.

[14] BioinformaticsAndMe, "Bioinformaticsandme, Bayesian theory, bioinformaticsandme blog," 2018. [Online]. Available: https://bioinformaticsandme.tistory.com/47.

[15] H. Herodotou, Y. Chen and J. Lu, "A survey on automatic parameter tuning for big data processing systems," *ACM Computing Surveys*, vol. 53, no. 43, pp. 1–37, 2020.

[16] K. Ji and Y. Kwon, "Map reduce performance optimization by changing hadoop parameter," in *Proc. of KIIT Conf.*, Cheongju, Korea, pp. 1–5, 2020.

[17] K. Ji and Y. Kwon, "Hadoop Map reduce performance optimization analysis by calibrating hadoop parameters," *Journal of KIIT*, vol. 19, no. 6, pp. 9–19, 2021.

[18] Y. Kim, H. Jeong, W. Choi, J. Kim, J. Choi *et al.,* "Hadoop performance analysis for distributed big-data processing in a virtualized cluster environment," in *Proc. of the Korean Information Science Society Conf.*, Daejeon, South Korea, vol. 39, pp. 13–15, 2012.

[19] S. Jeon, H. Chung, W. Choi, H. Shin, J. Chun *et al.,* "Map reduce tuning to improve distributed machine learning performance," in *IEEE First Int. Conf. on Artificial Intelligence and Knowledge Engineering (AIKE)*, Laguna Hills, CA, USA, pp. 198–200, 2018.

[20] H. Alkahtani, P. Gardner-Stephen and R. Goodwin, "A taxonomy of email spam filters," in *The Int. Arab Conf. on Information Technology*, Zarqa, Jordan, 2011.

[21] Wikipedia, "Bag-of-words_model_in_computer_vision," Wikimedia foundation, Inc, 2021. [Online]. Available: https://en.wikipedia.org/wiki/.

[22] D. Somvanshi, "A survey on spam filtering methods and Map-Reduce with SVM," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 3, pp. 490–494, 2017.

[23] T. Liu, W. Tsao and C. Lee, "A High-performance image-spam filtering system," in *Ninth Int. Symposium on Distributed Computing and Applications to Business, Engineering and Science*, Hong Kong, China, pp. 445–449, 2010.

[24] T. Vyas, P. Prajapati and S. Gadhwal, "A survey and evaluation of supervised machine learning techniques for spam e-mail filtering," in *2015 IEEE Int. Conf. on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India, 2015.

[25] J. Jeong, "MapReduce performance tuning for Big data," Korea: Gruter, 2012. [Online]. Available: http://www.gruter.com/technology/platform/.

[26] S. Sharma, "Advanced hadoop tuning and optimizations," 2009. [Online]. Available: https://www.slideshare.net/impetus Info/ppt-on-advanced-hadoop tuning-n-optimization.

[27] W. Sun, G. C. Zhang, X. R. Zhang, X. Zhang, N. N. Ge, "Fine-grained vehicle type classification using lightweight convolutional neural network with feature optimization and joint learning strategy," *Multimedia Tools and Applications*, vol. 80, no. 20, pp. 30803–30816, 2021.

[28] S. Murali, *Email-Spam-Classifier-Using-Naive-Bayes*, GitHub, 2018. [Online]. Available: https://github.com/Surya-Murali/Email-Spam-Classifier-Using-Naive-Bayes.

[29] M. Seltz, *W261-Fall2016*, Github, 2016. [Online]. Available: https://github.com/mseltz/W261-Fall2016/tree/master/Week02.

[30] G. Ko, *Mapreduce Compression*, Dev Blog, 2019. [Online]. Available: https://gunju-ko.github.io/hadoop/2019/11/29/Mapreduce-Compress.html.

[31] Tech Tutorials, 2018. [Online]. Available: https://www.netjstech.com/2018/04/how-to-compress-mapreduce-job-output-hadoop.html.

[32] H. Ciritoglu, L. Almeida, E. Almeida, T. Buda, J. Murphy *et al.,* "Investigation of replication factor for performance enhancement in the hadoop distributed file system," in *Int. Conf. on Performance Engineering (ICPE)'18 Companion*, Berlin, Germany, pp. 136–140, 2018.