Experimental Comparison to Improve Performance of Spam Filter System

- Using RNN with variety parameters and Optimaizers -

Jinha Hwang
Dept of Computer Engineering line, Kyung-Hee University
Yongin City, Republic of Korea
fkrlsp2@naver.com

ABSTRACT

In this paper, we apply various parameters and various optimizers to the spam filtering system using machine learning with RNN. We derive the appropriate results from the experiment and analyze the reasons for the results.

I. Introduction

Recently, interest in machine learning is growing. As a result of using machine learning, spam filtering system is one of the most used systems. This spam filtering system allows the user to determine whether the text is spam or not before reading it. Therefore, if you can improve the performance of this useful spam filtering system, you can trust the results of the spam filtering system more reliably. This performance enhancement will inhibit illegal advertising and increase user convenience. Therefore, in order to improve the performance of spam filtering, it is necessary to find the most optimal value by setting various algorithms and parameters. In this paper, we propose a spam filtering system, which is a useful spam filtering system, by applying various experiment environments and analyze the best results.

II. Background

A. Machine Learning

Machine learning is a field of artificial intelligence that refers to the field in which algorithms and techniques are developed that enable computers to learn. For example, machine learning can be trained to identify whether emails received are spam or not. The core of machine learning is in representation and generalization. Representation is an evaluation of data, and generalization is processing of data that is not yet known. This is also the field of computational learning theory. There are various applications of machine learning. Such spam filtering and character recognition are the best known examples of this.

B. RNN

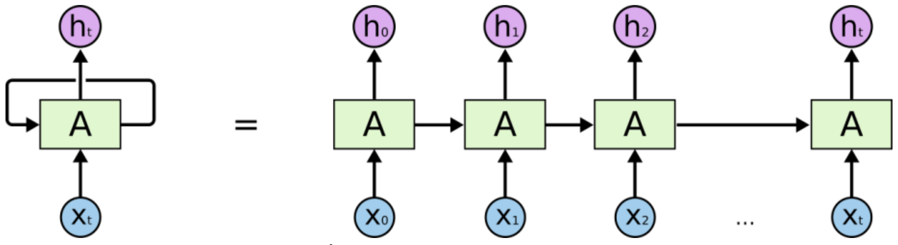


Figure 1 RNN Structure

A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

III. System Tested and Experiment Environment

This section describes the system environment and experimental environment used in this paper.

A. *System environment*

The system environment used in this experiment was used as follows.

- OS, Ubuntu 16.04 LTS
- CPU, 3.40Ghz, 1 core
- Memory 3GB
- Virtual Machine environment (VM Ware Work Station Pro)
- No GPU (CPU only mode)
- Using Tensorflow, Python, Matplotlib

B. *Dataset*

The data set used in this experiment was the SMS Spam Collection Data Set in UIC machine learning repository. The provided data sets have a total of 5574 instances and are provided in the following form [1].

```
ham   What you doing?how are you?
ham   Ok lar... Joking wif u oni...
ham   dun say so early hor... U c already then say...
ham   MY NO. IN LUTON 0125698789 RING ME IF UR AROUND! H*
ham   Siva is in hostel aha:-.
ham   Cos i was out shopping wif darren jus now n i called him 2 ask wat present he wan lor. Then he started guessing who i
was wif n he finally guessed darren lor.
spam   FreeMsg: Txt: CALL to No: 86888 & claim your reward of 3 hours talk time to use from your phone now!
ubscribe6GBP/ mnth inc 3hrs 16 stop?txtStop
spam   Sunshine Quiz! Win a super Sony DVD recorder if you canname the capital of Australia? Text MQUIZ to 82277. B
   spam   URGENT! Your Mobile No 07808726822 was awarded a L2,000 Bonus Caller Prize on 02/09/03! This is our 2nd
attempt to contact YOU! Call 0871-872-9758 BOX95QU
```

Textbox 2 Data example

The data can distinguish between normal and spam messages using “ham” and “spam” as follows, and there are no Missing Values. In addition, non-normal characters, abbreviations, and errors in data are frequently observed.

C. *Preprocessing*

Since the data is not refined, the data must undergo a preprocessing process.

```
def clean_text(text_string):
    text_string = re.sub(r'([^\s\w]|_|[0-9])+', '', text_string)
    text_string = " ".join(text_string.split())
    text_string = text_string.lower()
    return(text_string)
```

Textbox 2 Preprocessing

Using the regular expressions as follows, all the numbers except the characters were completely removed, followed by removing the consecutive whitespace, and then pre-processing all characters to be treated in lowercase.

Additionally, the dataset being handled does not include the Missing Value, so the fill process is skipped.

Data to be used for training and data to be used for testing were divided into 8: 2.

D. *Learning Parameter*

The Default Machine learning parameters used in the experiment are set as follows.

```
# Set RNN parameters
                       epochs = 20
                       batch_size = 250
                       max_sequence_length = 25
                       rnn_size = 10
                       embedding_size = 50
                       min_word_frequency = 10
                       learning_rate = 0.0005
                       dropout_keep_prob = 0.5
                       optimizer = adamOptimizer
```

Textbox 3 RNN Learning Parameters

IV. EXPERIMENT

The code used in the experiment was open source, which was published in the **tensorflow cookbook**. [1] We constructed the RNN network by using this source code and experimented with the code modified for the purpose of the experiment.

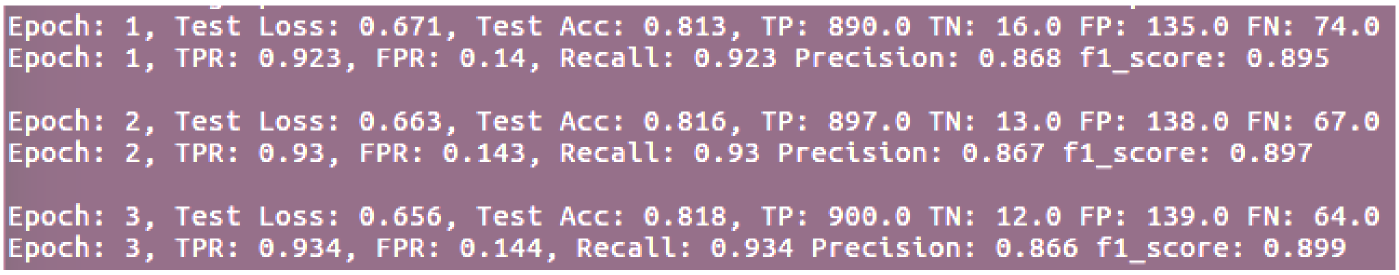
In this experiment, we proceed while converting RNN parameters and optimizers. In this case, the change value used in the experiment converts the values of the epoch, the running rate, and the optimizer.

Table 1 Parameters

epoch	10	20	50	100
learning rate	0.0005	0.005	0.05	0.5
optimizer	GradientDescent Optimizer	AdamOptimizer	RMSProp Optimizer	

The changes in parameters used in the experiment are as shown above.

The average value is calculated by **repeating 20 times** for each process to produce a more uniform result every time one option is given. In this experiment, 200 times were repeated.



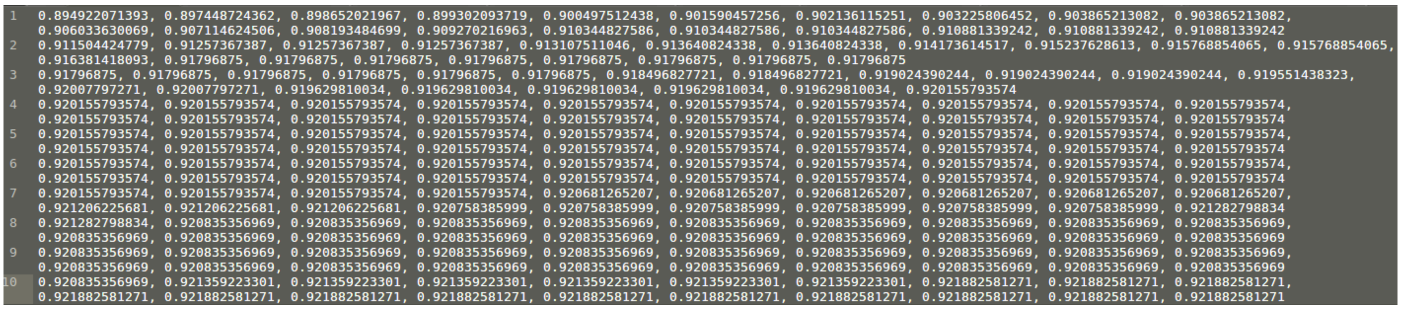
```
Epoch: 1, Test Loss: 0.671, Test Acc: 0.813, TP: 890.0 TN: 16.0 FP: 135.0 FN: 74.0
Epoch: 1, TPR: 0.923, FPR: 0.14, Recall: 0.923 Precision: 0.868 f1_score: 0.895

Epoch: 2, Test Loss: 0.663, Test Acc: 0.816, TP: 897.0 TN: 13.0 FP: 138.0 FN: 67.0
Epoch: 2, TPR: 0.93, FPR: 0.143, Recall: 0.93 Precision: 0.867 f1_score: 0.897

Epoch: 3, Test Loss: 0.656, Test Acc: 0.818, TP: 900.0 TN: 12.0 FP: 139.0 FN: 64.0
Epoch: 3, TPR: 0.934, FPR: 0.144, Recall: 0.934 Precision: 0.866 f1_score: 0.899
```

Figure 3 Code output example

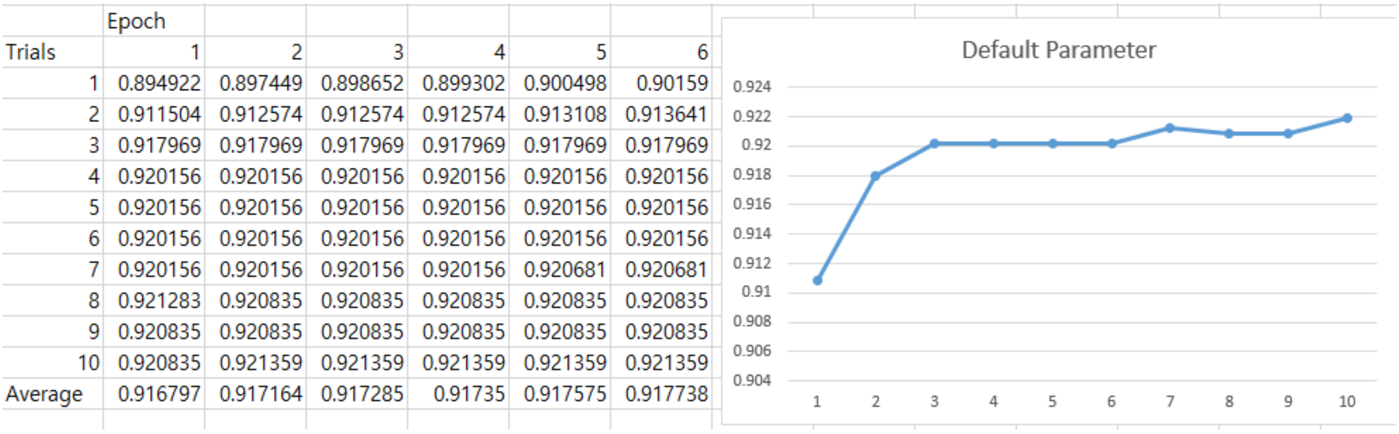
The above values are generated as an example when producing the output through the default parameters.



```
0.894922071393, 0.897448724362, 0.898652021967, 0.899302093719, 0.900497512438, 0.901590457256, 0.902136115251, 0.903225806452, 0.903865213082, 0.903865213082,
0.906033630069, 0.907114624506, 0.908193484699, 0.909270216963, 0.910344827586, 0.910344827586, 0.910344827586, 0.910881339242, 0.910881339242, 0.910881339242
0.911504424779, 0.91257367387, 0.91257367387, 0.91257367387, 0.913107511046, 0.913640824338, 0.913640824338, 0.914173614517, 0.915237628613, 0.915768854065, 0.915768854065,
0.916381418093, 0.91796875, 0.91796875, 0.91796875, 0.91796875, 0.91796875, 0.91796875, 0.91796875, 0.91796875, 0.91796875
0.91796875, 0.91796875, 0.91796875, 0.91796875, 0.91796875, 0.91796875, 0.918496827721, 0.918496827721, 0.919024390244, 0.919024390244, 0.919024390244, 0.919551438323,
0.92007797271, 0.92007797271, 0.919629810034, 0.919629810034, 0.919629810034, 0.919629810034, 0.919629810034, 0.920155793574
0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574,
0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574
0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574,
0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574
0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574,
0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574
0.920155793574, 0.920155793574, 0.920155793574, 0.920155793574, 0.920681265207, 0.920681265207, 0.920681265207, 0.920681265207, 0.920681265207, 0.920681265207,
0.921206225681, 0.921206225681, 0.921206225681, 0.920758385999, 0.920758385999, 0.920758385999, 0.920758385999, 0.920758385999, 0.920758385999, 0.921282798834
0.921282798834, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969,
0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969
0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969,
0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969, 0.920835356969
0.920835356969, 0.921359223301, 0.921359223301, 0.921359223301, 0.921359223301, 0.921359223301, 0.921359223301, 0.921882581271, 0.921882581271, 0.921882581271,
0.921882581271, 0.921882581271, 0.921882581271, 0.921882581271, 0.921882581271, 0.921882581271, 0.921882581271, 0.921882581271, 0.921882581271, 0.921882581271
```

Figure 4 Txt File output

To save the data using this result, we first stored it in the txt file. At this time, the stored value was stored as the **F1 measure** value to measure the performance of the comprehensive test. The separator for each epoch is "," and the separator for each number of attempts is separated by a line break.



	Epoch					
Trials	1	2	3	4	5	6
1	0.894922	0.897449	0.898652	0.899302	0.900498	0.90159
2	0.911504	0.912574	0.912574	0.912574	0.913108	0.913641
3	0.917969	0.917969	0.917969	0.917969	0.917969	0.917969
4	0.920156	0.920156	0.920156	0.920156	0.920156	0.920156
5	0.920156	0.920156	0.920156	0.920156	0.920156	0.920156
6	0.920156	0.920156	0.920156	0.920156	0.920156	0.920156
7	0.920156	0.920156	0.920156	0.920156	0.920681	0.920681
8	0.921283	0.920835	0.920835	0.920835	0.920835	0.920835
9	0.920835	0.920835	0.920835	0.920835	0.920835	0.920835
10	0.920835	0.921359	0.921359	0.921359	0.921359	0.921359
Average	0.916797	0.917164	0.917285	0.91735	0.917575	0.917738

Figure 5 Excel sheet

Use Excel, one of the spreadsheet programs, to effectively handle and analyze the F1 measure values stored in this text file. This Excel provides useful functions to manage data and powerful function to visualize through chart composition, so it can be used effectively in this paper.

V. EXPERIMENT RESULT

A. *Epoch*

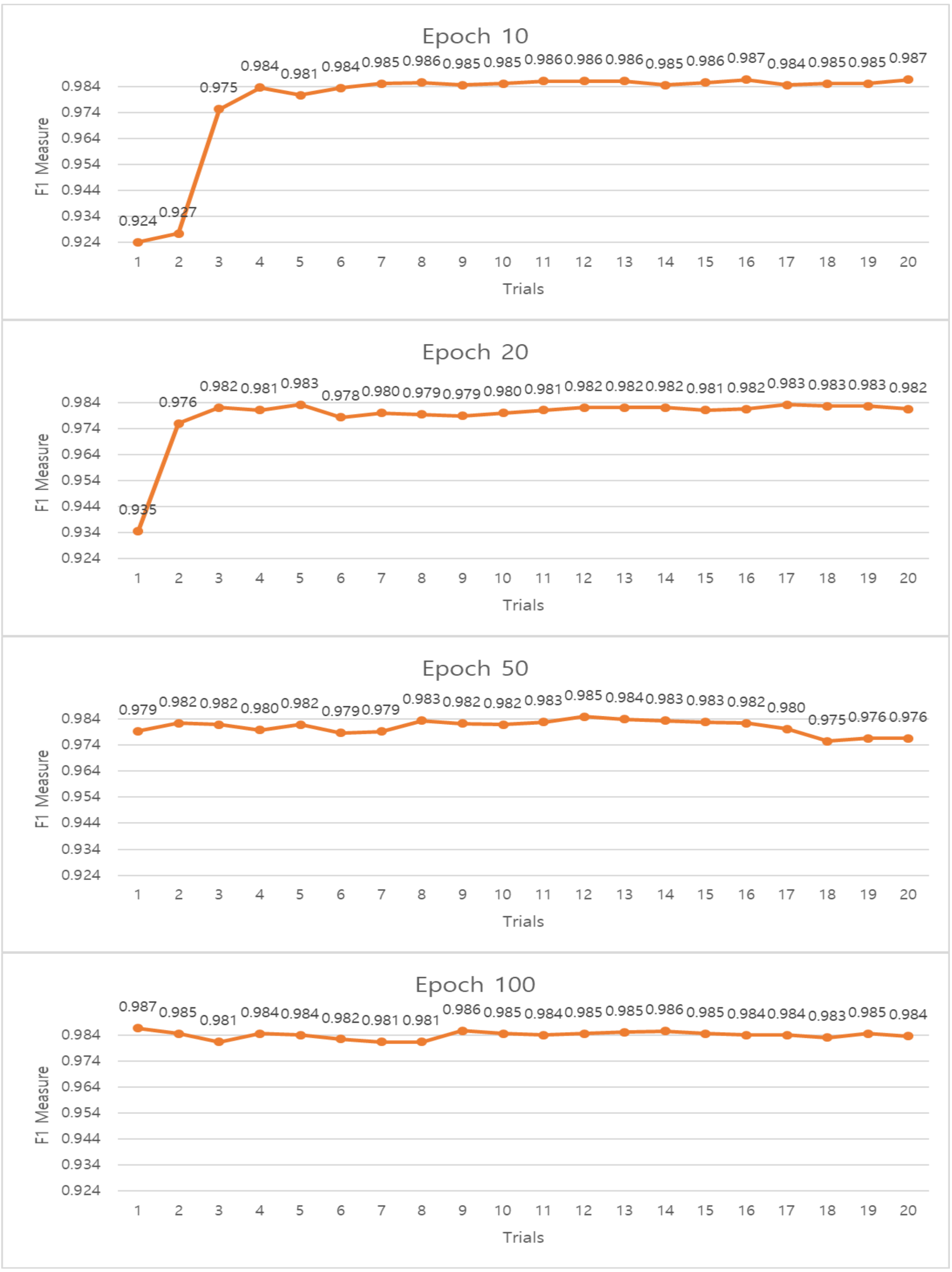


Figure 6 Epoch Chart

B. *Learning rate*

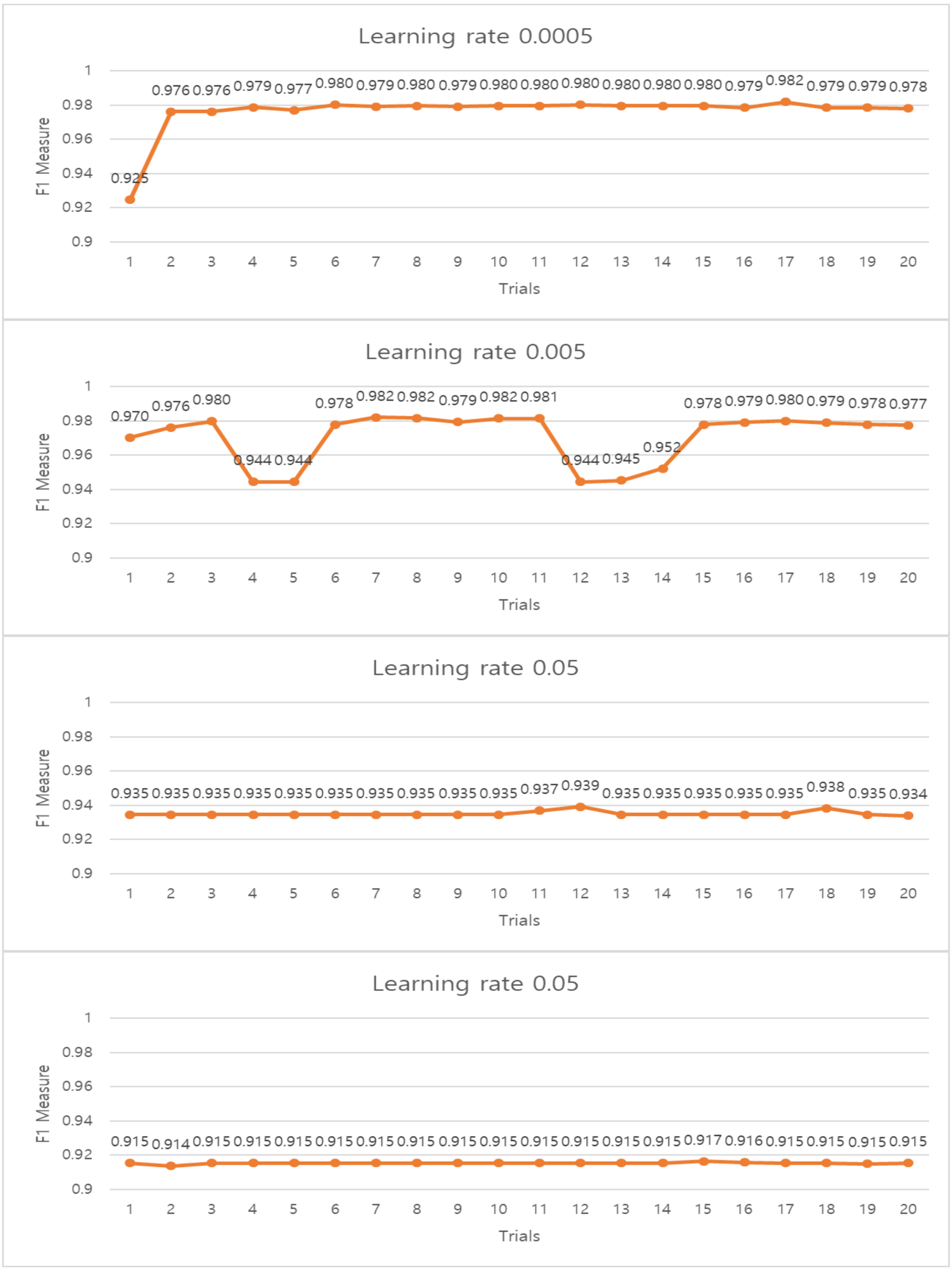


Figure 7 Learning rate Chart

C. *Optimizer*

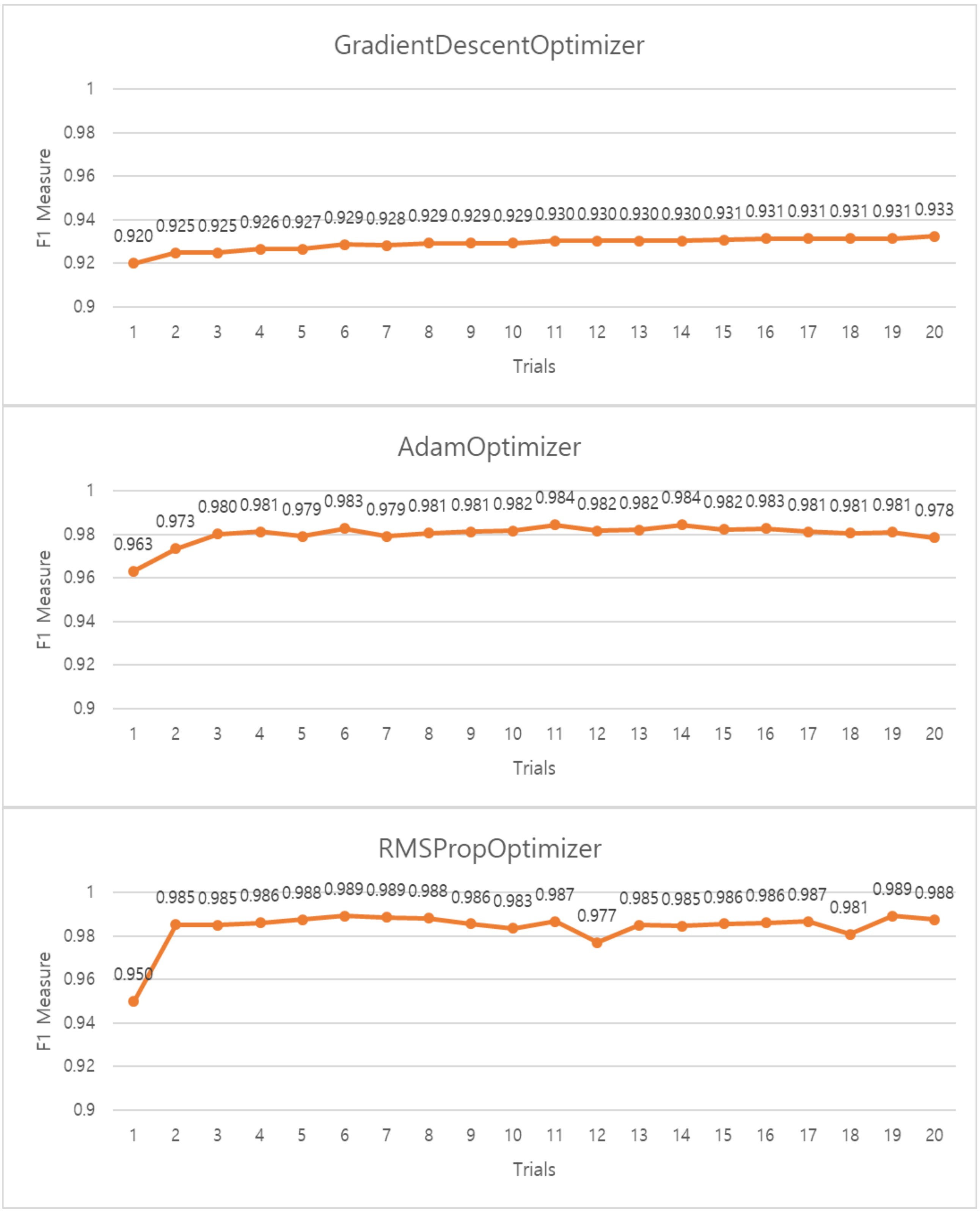


Figure 8 Optimizer Chart

D. Result

The experiment that was carried out while increasing the echo was aimed at confirming the offer and under fitting in the machine learning through the CNN network using the data set. However, unexpectedly, all epochs showed appropriate results. The results of all experimental results showed no significant difference from the mean value of 0.9852.

Experiments conducted with increasing running rate showed useful results. Experiments on the running rate were intended to observe the results when the overshooting and running rate were too small to reach the optimal value when the running rate was too large. The best result was obtained at the running rate of 0.0005 which is the minimum, and it was confirmed that the measured value of F1 Measure was significantly decreased as the running rate increased. This seems to be an overshoot phenomenon that cannot find an optical value due to a large running rate.

Experimenting with changing the optimizer was an experiment to see which optimizer would be good. “GradientDescentOptimizer” did not show good overall results. Overall, AdamOptimizer and “RMSPropOptimizer” showed better results than “GradientDescentOptimizer”. The best performance in this experiment was running with “RMSPropOptimizer” but it did not show much difference from “AdamOptimizer”. Later, when you do a text classification similar to the spam filter in the experiment, you can choice it to your taste.

VI. CONCLUSION

Through this experiment, spam filtering was performed through RNN network. Experiments were carried out by applying various experimental parameters such as epoch, running rate, and optimizer. Through this experiment, we were able to know what parameter values should be used when applying machine learning using the RNN network for the spam data set. If you use the results to approach other problems using text classification, you will be able to reduce the effort of finding the optimal value.

The **source code** and data for the experiment The **text file** and the data chart The **Excel file** can be checked in my **Github** [2]

REFERENCES

[1] https://github.com/nfmcclure/tensorflow_cookbook/blob/master/09_Recurrent_Neural_Networks/02_Implementing_RNN_for_Spam_Prediction/02_implementing_rnn.

[2] https://github.com/Phigaro/Experimental-Comparison-to-Improve-Performance-of-Spam-Filter-System