

Header Based Spam Filtering Using Machine Learning Approach

M Mubasher Hassan

Dept. of ITE, BGSB University Rajouri (J&K), India.

Dr. Mirza Waseem Hussain

P.G. Department of Computer Sciences, University of Kashmir, North Campus, India.

Abstract – Email is one of the fastest, easiest and cost effective means of communication. However, spam not only wastes our precious and limited resources, but also possess a security risk; therefore, there should be an effective means by which we can avoid this nuisance. In this paper, an effective machine learning approach for filtering the spam is proposed, by identifying potential header and body fields. The Machine learning dataset used is combination of open source spambase and custom developed dataset for header fields. Weka in combination with naïve-bayes classification technique is applied using machine learning based dataset encompassing certain header field based attributes. A mail client is designed in asp.net, which is used as an interface that separates the header and body of the mail. The effectiveness of the proposed machine learning approach was compared with one that does not use machine learning approach and it was found the proposed approach performed better when using machine-learning approach.

Index Terms – Weka, Naïve Bayesian, Machine learning, Spam, Email Header, Email Body, spambase.

1. INTRODUCTION

With the revolution in the technology, particularly in telecommunication, there has been an enormous shift the way do business and communicate. The internet is acting as a lifeline of the current world, with most of the communication being done via email. In a report published by Radicati group, it was estimated that 3.7 billion users would avail email services by the end of 2017, accounting to 269 billion mails per day [1] [2]. About 49.7 % of emails are spam mails [1]. Spam is defined as unsolicited, unwanted or irrelevant bulk messages sent to the user for phishing, advertisement or spreading malware, etc. [3] [4] [5] [6]. Spam mail consumes considerable amount of our vital resources in terms of not only bandwidth and money, but also wastes our precious time [7]. Now a day's spammer uses sophisticated techniques to bypass spam-filtering process.

The email consists of two parts; header and body section. The header section is unique to each mail, comprising summarized report of sender and receiver. The body consists of the actual data to be delivered. Both header and body section holds valuable information utilized to differentiate between spam and

non-spam email. There are various methods by which we can identify legality of the mail, which are as follows [8]:

- Blacklist/Whitelist method.
- Mail Header Checking.
- Bayesian analysis.
- Keyword checking.

In addition to above, identification techniques there are various spam filtering techniques that are as follows [8]:

- Rule based filtering [9].
- Distributed adaptive blacklists [10].
- Bayesian classifier [11].
- Naïve Bayes Spam Filtering [12].
- K nearest neighbour [13, 14].
- Support vector machine (SVM) [15, 16].
- Technique of search engines [17].
- Technique of genetic engineering [18].
- Technique of artificial immune system [19].
- Fighting Spammer's Resources Approach [20].
- Artificial Neural Networks Approach [21].

It has been observed that spam filtering based on body section is very time consuming and rigorous. Furthermore, the existing techniques for filtering the spams are inefficient with most of the filtering techniques concentrating only on the body section of the email message.

Machine learning can be a good candidate for filtering the spams due to their inbuilt properties like; capability to learn, fault tolerant, able to draw an implicit relationship between the variables, etc. We propose an intelligent technique based on Naïve Bayesian classification using custom data sets in combination to existing datasets for classifying spam and non-spam emails. The results confirm that we were able to make better predictions than contemporary spam filtering techniques.

2. LITERATURE REVIEW

Spamming is one of the major problem faced now a day's by email users. Various researchers came up with different approaches/solutions. Broadly, the solution to filter spamming

can be classified into two categories; Knowledge engineering and Machine learning [8]. Knowledge engineering uses a set of rules to distinguish between spam and non-spam mails. However, this technique has some limitations, as the rules need to be continuously updated. On the other hand, machine-learning approach requires no set of predefined rules but uses training examples or training data set to learn for predicting whether mail it is spam or non-spam.

Sahami et.al [22] in 1998 were the first to propose spam-filtering technique based on Naïve Bayes classifiers. The combination of message phrasal words (free tour, free money, and lottery) and non-phrasal rules derived from message header were used. The experiment was performed on two private corpora, with results confirming better results by low true false positive rates.

[23] extended the work of Sahami et.al by applying classifiers on text, relying heavily on the assumption of naïve independence, that is all the features used are statically independent. The proposed technique performed faster as compared to Sahami et.al approach.

Li and Zhang[24] proposed a spam filtering technique using approximate classification making this technique even much faster as compared to previous one with greater accuracy.

Ensemble learning in combination with decision tree are employed to classify spam emails with the accuracy of 94.2% [25].

[26] proposes a classification technique to filter the spam emails by assigning some weights to the features extracted from message body. In addition to this proposed technique tries to reduce the dimensions to increase the efficiency of the proposed technique. Results also confirm performance improvement.

3. PROPOSED WORK

Considering the existence of scope to improve the spam filtering activity we have started the research work around spam filtering based on e-mail headers complimented with machine learning approach rather than the age old techniques like listing methods (blacklist, whitelist or grey list).

An application platform capable of sending and receiving test mails has been developed using .NET (C#). Weka is used to train and Naïve Bayes classifier for the purposes of spam detection .

Machine learning dataset spambase created by Mark Hopkins, Erik Reeber, George Forman, Jaap Suermondt, Hewlett-Packard Labs offered by UCI- machine learning repository is a collection of e-mails (blend of spam and non spam) from the postmaster and the people who reported spams. The dataset

usually, is used as ARFF file and contains various attributes based on E-mail body. Most of the attributes indicate whether a particular word or character was frequently occurring in the e-mail.

The work is based on Naïve Bayes Classification technique, which includes machine learning based filtering done not only on the message body but also on some chosen fields in e-mail headers. Proposed methodology will use custom developed dataset for header fields and an open source dataset spambase for e-mail body dataset testing. To train a naive Bayes model, we can use fitcnb. After training a naive Bayes model, we can predict labels or estimate posterior probabilities by passing the model and predictor data to predict as shown in Fig. 1.

The body spambase.arff is an open source dataset and the header training dataset is created manually from the attributes of legitimate and spam mails respectively. These training datasets are used in weka for classification purposes. A mail client is designed in asp.net which is used as an interface that separates the header and body of the mail. On the header of the mail name and value pairs are separated according to the attributes on which we are testing our spam filter and on the body of the mail tokenization is done according to the attributes written in spambase.arff. This process gives us our testing datasets. Then these four datasets i.e body training and testing datasets and header training and testing datasets are given as attributes to the function, which is connected to weka through a java instance.

The potential feature set that we extracted from this header fields includes features like:

- Total time.
- Number of receivers (N).
- Total recipients.
- IP and domain name validity.
- Date and time validity.

Weka does classification according to these datasets and return the calculations predicting whether the mail is legitimate or spam. The coding is done in MATLAB for functions, which create these test datasets. Spambase has been studied and converted to a format suitable for integration in a MATLAB platform to take advantage of MATLAB's Strong visualization and analysis features. MATLAB is connected to weka through java instance as weka is designed in java. These datasets are moved from MATLAB to weka by these java instances and through these instances the calculations from weka are displayed in MATLAB. From these calculations we determine whether our approach have improved spam filtering or not. We have used weka as our classifier because it calculates various values that would determine the spamicity of a mail.

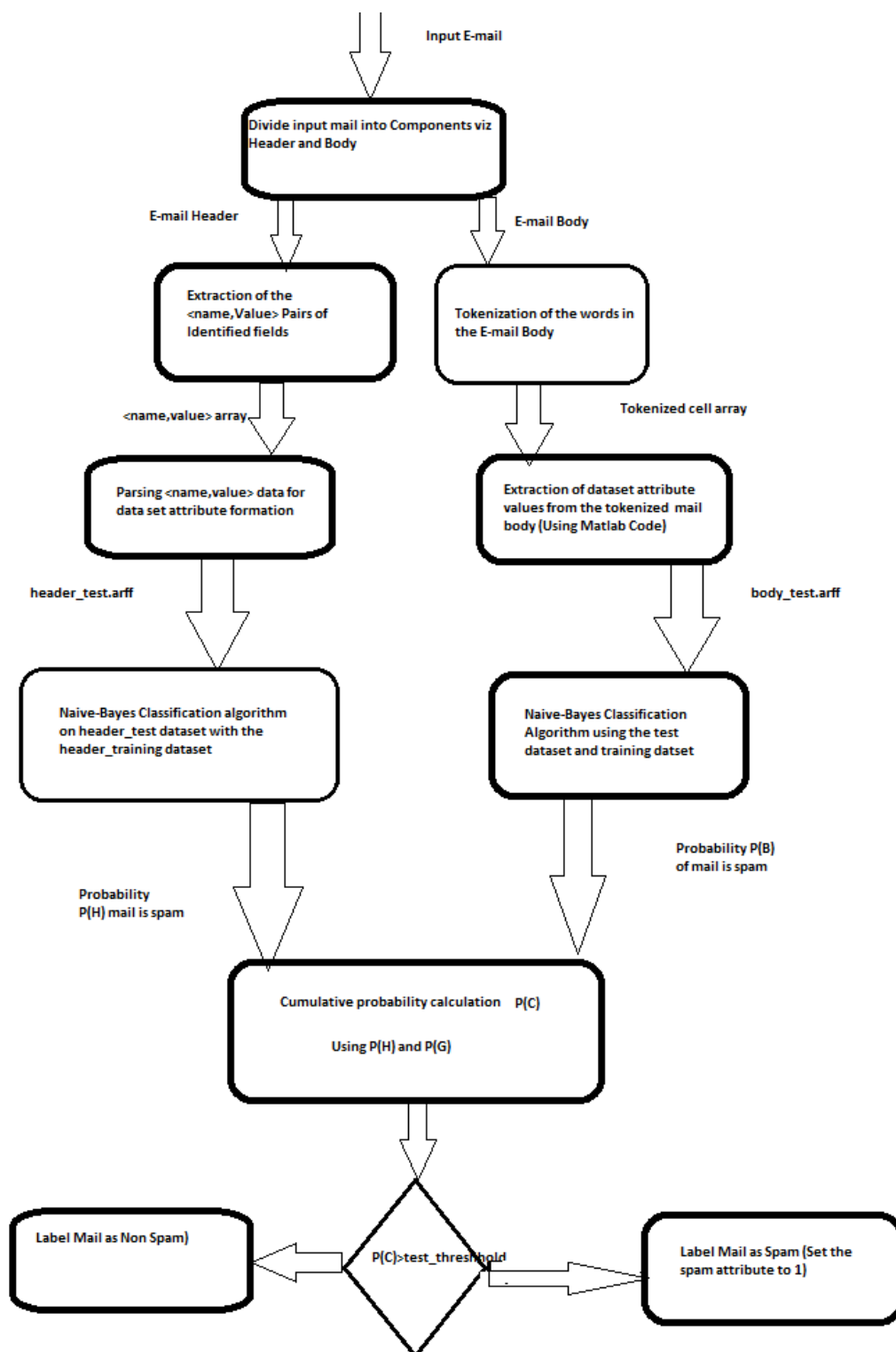
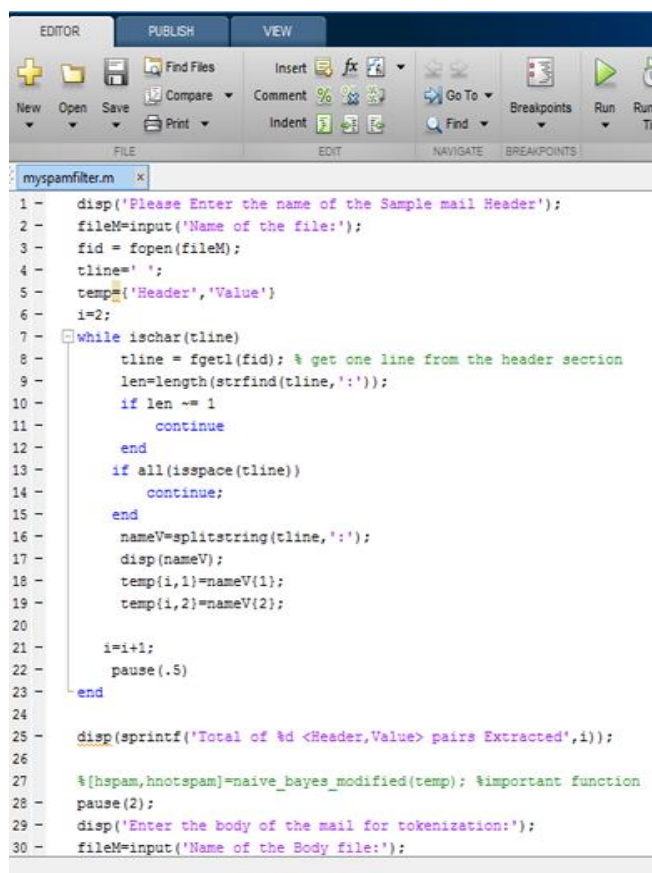


Figure 1 Flowchart of proposed spam filtering approach.

Our interface in MATLAB is myspamfilter.m program as shown in Fig. 2. In this program, we first extract the header value pairs in the header of our mail by making ':' as delimiter. Splitstring() function is used to split the header attributes. Then we tokenize the body of the mail according to the words described in spambase.arff file. We count the word frequency using arff_body_parser() function. This program gives us an array of data or feature_set, which is stored in our testspambase.arff file for further classification.

After header value separation of header attributes and tokenization of words in body of the mail, two files that are test.arff and training.arff are made arguments to the function myclassifier(a_test,a_training) which classifies the test data using naïve bayes algorithm and predicts that whether the mail is spam or no as shown in Fig. 3. This classification is done through weka using wekaClassify() function. As described weka is an open source classification application whose input is an arff file i.e attribute relation file format. Weka returns various values like mean, standard deviation, error rate etc. and also predicts whether a mail is spam or not.

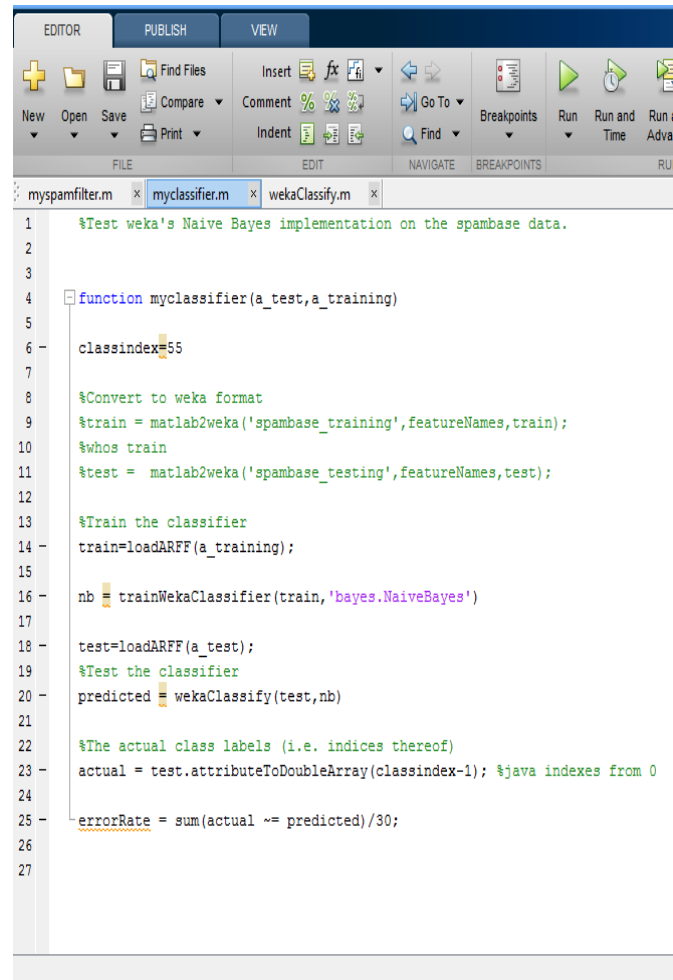


```

1  disp('Please Enter the name of the Sample mail Header');
2  fileM=input('Name of the file:');
3  fid = fopen(fileM);
4  tline= ' ';
5  temp={'Header','Value'};
6  i=2;
7  while ischar(tline)
8      tline = fgetl(fid); % get one line from the header section
9      len=length(strfind(tline,':'));
10     if len ~= 1
11         continue
12     end
13     if all(isspace(tline))
14         continue;
15     end
16     nameV=splitstring(tline,':');
17     disp(nameV);
18     temp(i,1)=nameV(1);
19     temp(i,2)=nameV(2);
20
21     i=i+1;
22     pause(.5)
23 end
24
25 disp(sprintf('Total of %d <Header,Value> pairs Extracted',i));
26
27 % [hspam,hnotspam]=naive_bayes_modified(temp); %important function
28 pause(2);
29 disp('Enter the body of the mail for tokenization:');
30 fileM=input('Name of the Body file:');

```

Figure 2 myspamfilter.m



```

1  %Test weka's Naive Bayes implementation on the spambase data.
2
3
4  function myclassifier(a_test,a_training)
5
6      classindex=55
7
8      %Convert to weka format
9      %train = matlab2weka('spambase_training',featureNames,train);
10     %whos train
11     %test = matlab2weka('spambase_testing',featureNames,test);
12
13     %Train the classifier
14     train=loadARFF(a_training);
15
16     nb = trainWekaClassifier(train,'bayes.NaiveBayes')
17
18     test=loadARFF(a_test);
19     %Test the classifier
20     predicted = wekaClassify(test,nb)
21
22     %The actual class labels (i.e. indices thereof)
23     actual = test.attributeToDoubleArray(classindex-1); %java indexes from 0
24
25     errorRate = sum(actual ~= predicted)/30;
26
27

```

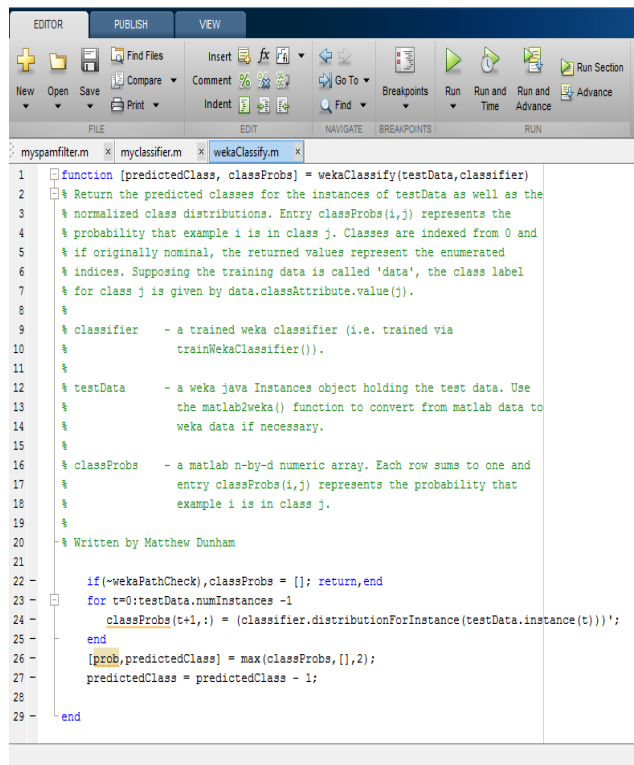
Figure 3 myclassifier.m

wekaClassify(testData,classifier) as shown in Fig. 4 is the actual function which gets the predicted probability and predicted class from weka by giving it test.arff and training object. The values are calculated in weka but are displayed in MATLAB.

loadARFF() function as shown in Fig. 5 is used to load data from a weka .arff file into a java weka instances object for use by weka classes. This can be converted for use in MATLAB by passing wekaOBJ to the weka2matlab function.

matlab2weka() function as shown in Fig. 6 is used to convert MATLAB data to a weka java instances object for use by weka classes.

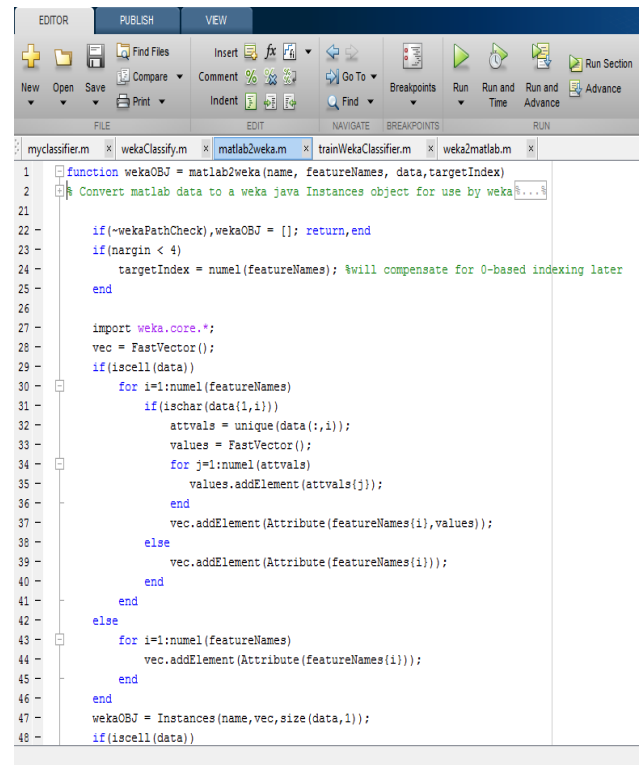
weka2matlab() function as shown in Fig. 7 converts weka data, stored in a java weka instances object to a MATLAB.



```

1 function [predictedClass, classProbs] = wekaClassify(testData, classifier)
2 % Return the predicted classes for the instances of testData as well as the
3 % normalized class distributions. Entry classProbs(i,j) represents the
4 % probability that example i is in class j. Classes are indexed from 0 and
5 % if originally nominal, the returned values represent the enumerated
6 % indices. Supposing the training data is called 'data', the class label
7 % for class j is given by data.classAttribute.value(j).
8 %
9 % classifier - a trained weka classifier (i.e. trained via
10 % trainWekaClassifier()).
11 %
12 % testData - a weka java Instances object holding the test data. Use
13 % the matlab2weka() function to convert from matlab data to
14 % weka data if necessary.
15 %
16 % classProbs - a matlab n-by-d numeric array. Each row sums to one and
17 % entry classProbs(i,j) represents the probability that
18 % example i is in class j.
19 %
20 % Written by Matthew Dunham
21
22 if(~wekaPathCheck), classProbs = []; return, end
23 for t=0:testData.numInstances-1
24     classProbs(t+1,:) = (classifier.distributionForInstance(testData.instance(t)))';
25 end
26 [prob, predictedClass] = max(classProbs, [], 2);
27 predictedClass = predictedClass - 1;
28
29 end
  
```

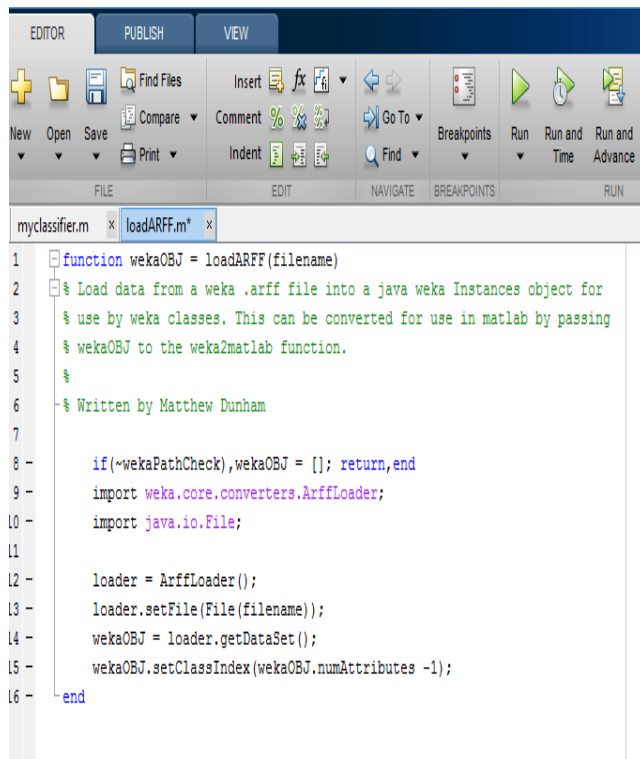
Figure 4 wekaClassify.m



```

1 function wekaOBJ = matlab2weka(name, featureNames, data, targetIndex)
2 % Convert matlab data to a weka java Instances object for use by weka
3
21
22 if(~wekaPathCheck), wekaOBJ = []; return, end
23 if(nargin < 4)
24     targetIndex = numel(featureNames); %will compensate for 0-based indexing later
25 end
26
27 import weka.core.*;
28 vec = FastVector();
29 if(iscell(data))
30     for i=1:numel(featureNames)
31         if(ischar(data{1,i}))
32             attvals = unique(data{:,i});
33             values = FastVector();
34             for j=1:numel(attvals)
35                 values.addElement(attvals{j});
36             end
37             vec.addElement(Attribute(featureNames{i}, values));
38         else
39             vec.addElement(Attribute(featureNames{i}));
40         end
41     end
42 else
43     for i=1:numel(featureNames)
44         vec.addElement(Attribute(featureNames{i}));
45     end
46 end
47 wekaOBJ = Instances(name, vec, size(data,1));
48 if(iscell(data))
  
```

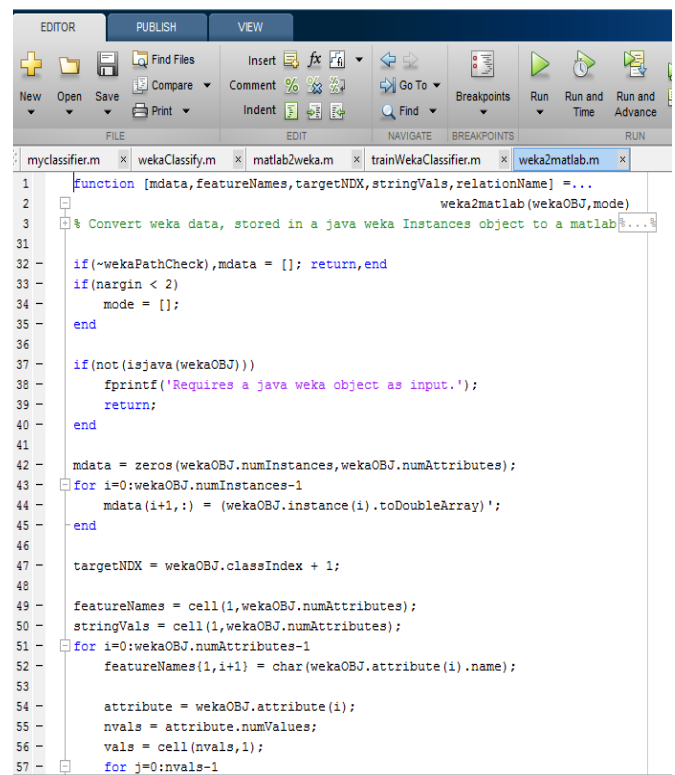
Figure 6 matlab2weka.m



```

1 function wekaOBJ = loadARFF(filename)
2 % Load data from a weka .arff file into a java weka Instances object for
3 % use by weka classes. This can be converted for use in matlab by passing
4 % wekaOBJ to the weka2matlab function.
5 %
6 % Written by Matthew Dunham
7
8 if(~wekaPathCheck), wekaOBJ = []; return, end
9 import weka.core.converters.ArffLoader;
10 import java.io.File;
11
12 loader = ArffLoader();
13 loader.setFile(File(filename));
14 wekaOBJ = loader.getDataSet();
15 wekaOBJ.setClassIndex(wekaOBJ.numAttributes - 1);
16
17 end
  
```

Figure 5 loadARFF.m



```

1 function [mdata, featureNames, targetNDX, stringVals, relationName] = ...
2     weka2matlab(wekaOBJ, mode)
3 % Convert weka data, stored in a java weka Instances object to a matlab
4
31
32 if(~wekaPathCheck), mdata = []; return, end
33 if(nargin < 2)
34     mode = [];
35 end
36
37 if(not(isjava(wekaOBJ)))
38     fprintf('Requires a java weka object as input. ');
39     return;
40 end
41
42 mdata = zeros(wekaOBJ.numInstances, wekaOBJ.numAttributes);
43 for i=0:wekaOBJ.numInstances-1
44     mdata(i+1,:) = (wekaOBJ.instance(i).toDoubleArray)';
45 end
46
47 targetNDX = wekaOBJ.classIndex + 1;
48
49 featureNames = cell(1, wekaOBJ.numAttributes);
50 stringVals = cell(1, wekaOBJ.numAttributes);
51 for i=0:wekaOBJ.numAttributes-1
52     featureNames{1,i+1} = char(wekaOBJ.attribute(i).name);
53
54     attribute = wekaOBJ.attribute(i);
55     nvals = attribute.numValues;
56     vals = cell(nvals, 1);
57     for j=0:nvals-1
  
```

Figure 7 weka2matlab.m

4. RESULTS AND DISCUSSIONS

The comparison of the proposed filtering technique with contemporary spam filtering approach was carried out in weka using matlabplaform in a passive manner. Further, MATLAB was used to calculate and compare the confusion matrix in both the scenarios. The matrix is based on set of test dataset (ARFF File) made up of 50 mails (1:1 ratio). The Matrix suggests some amount of evidence of performance enhancement as shown in table 1, table 2, Fig. 8, Fig. 9. Further, the significant amount of time was saved to suggest that header based machine learning can reduce the time required to classify the mails. The subsequent integration with a Mail User Agent could be done to test the performance in the active manner.

	Actual	
Prediction	Spam	Non-Spam
Spam	18	4
Non-Spam	7	21

Table 1 When only spambase dataset was used.

	Actual	
Predication	Spam	Non-Spam
Spam	19	3
Non-Spam	6	22

Table 2 When simplified spambase was used with custom header based dataset

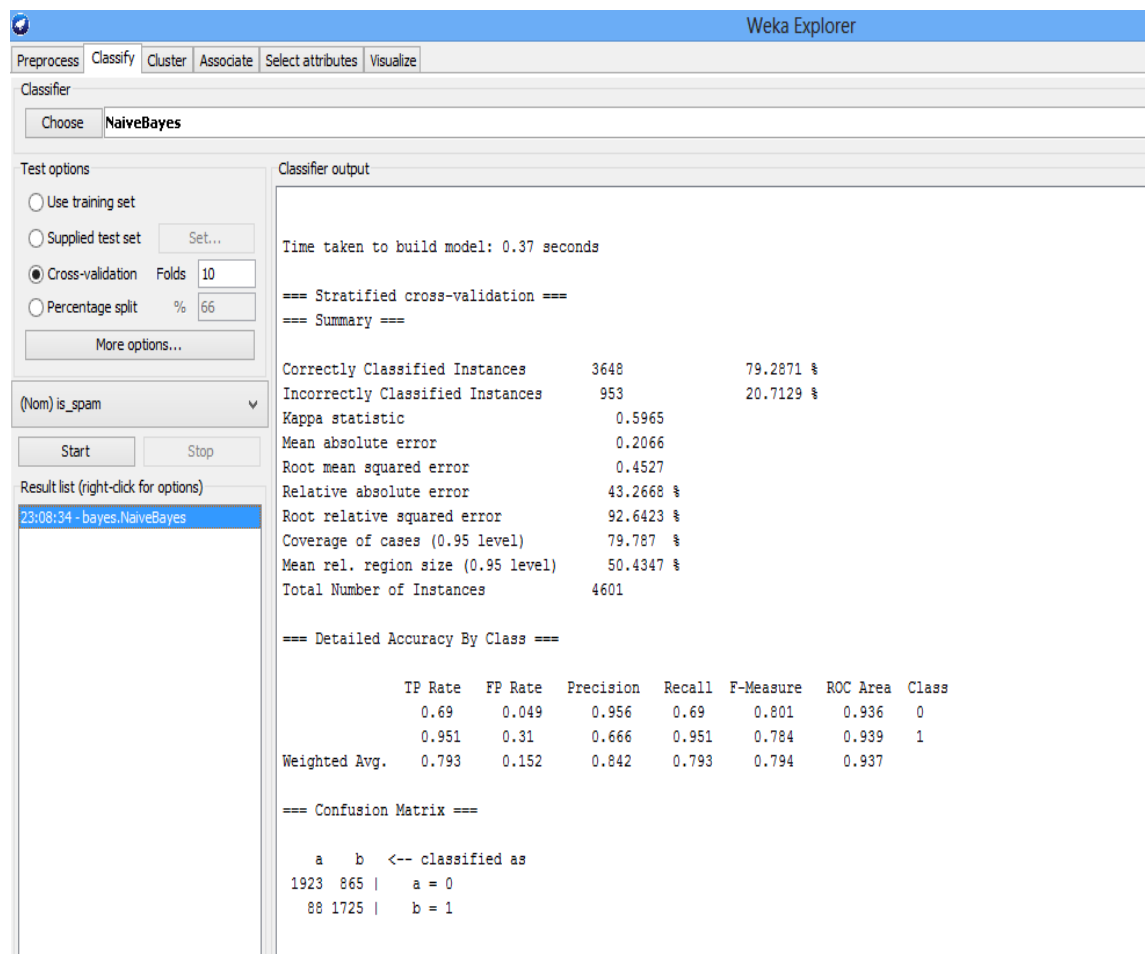


Figure 8 Performance in weka

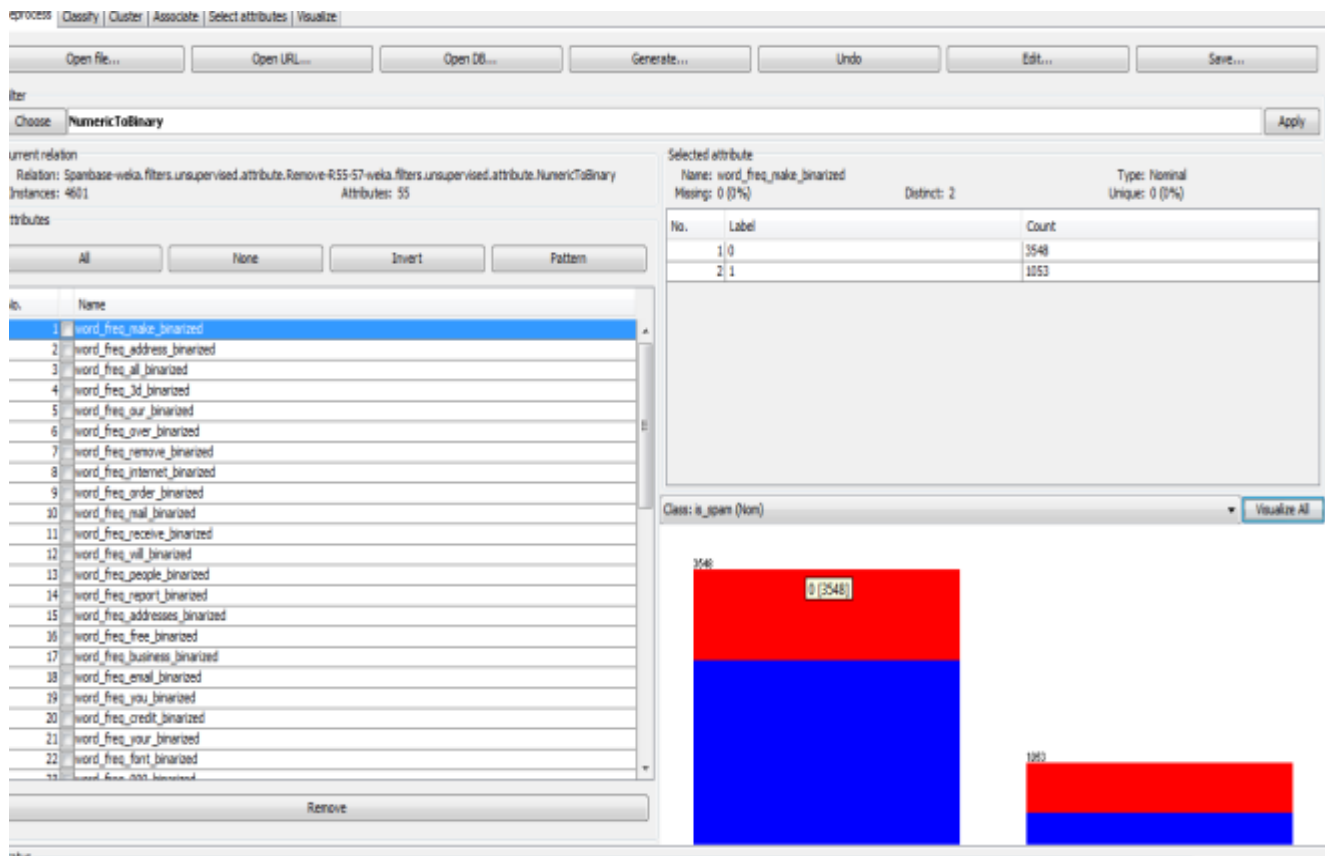


Figure 9 Various attributes of the spambase.arff as seen in the weka tool. The plot shows the percentage of spam and non-spam mails in the data set.

5. CONCLUSION

When the naïve-bayes classification technique is applied using machine learning based dataset encompassing certain header field based attributes, the overall improvement in spam filtering is achieved. Thus, the rigorous and time-consuming filtering done on e-mail body can be simplified without affecting the spam filter performance. This conclusion was arrived after developing training and testing datasets using the features of e-mail headers and incorporating it with spambase dataset.

REFERENCES

- [1] How Many Emails Are Sent Every Day. Retrieved from: <https://www.lifewire.com/how-many-emails-are-sent-every-day-1171210>
- [2] Email Statistics Report, "2015-2019 – Executive Summary," Retrieved from: <http://www.radicati.com>
- [3] The Definition of Spam. Retrieved from: <https://www.spamhaus.org/consumer/definition/>
- [4] What is spam? Retrieved from: <https://kb.iu.edu/d/afne>
- [5] Androutsopoulos I., Koutsias J., Chandrino K. V., and Spyropoulos C. D., "An experimental comparison of naïve Bayesian and keyword-based anti-spam filtering with personal e-mail messages," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, July 2000, pp. 160-167.
- [6] Androutsopoulos I., Paliouras G., Karkaletsis V., Sakkis G., Spyropoulos C. D., and Stamatopoulos P., (2000). "Learning to filter spam e-mail: A comparison of a naïve bayesian and a memory-based approach," 2000, arXiv preprint cs/0009009.
- [7] Tran. M., and Armitage. G., "End-users' resource consumption of spam and a 3D anti-spam evaluation framework," in *TENCON 2005 2005 IEEE Region 10*, November 2005, pp. 1-6.
- [8] Christina V. Karpagavalli S and Suganya G., "Article:A Study on Email Spam Filtering Techniques," in *International Journal of Computer Applications*, December 2010, Vol.12(1), pp. 7–9.
- [9] Fiumara G., Marchi M., Pagano R., and Provetti A., "Rule-Based Spam E-mail Annotation," in *P. Hitzler & T. Lukasiewicz (Eds.), Web Reasoning and Rule Systems: Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings, Berlin, Heidelberg: Springer Berlin Heidelberg,* 2010, pp.231-234. DOI:10.1007/978-3-642-15918-3_21
- [10] Yuxin Meng and Lam-For Kwok, 2014. "Adaptive blacklist-based packet filter with a statistic-based approach in network intrusion detection," in *J. Netw. Comput. Appl.*, March 2014, Vol. 39, pp. 83-92. DOI: <http://dx.doi.org/10.1016/j.jnca.2013.05.009>
- [11] Obied A., "Bayesian Spam Filtering", Department of Computer Science University of Calgary, 2007.
- [12] Tianda Yang, Kai Qian, Dan Chia-Tien Lo, K. Al Nasr and Ying Qian, "Spam filtering using Association Rules and Naïve Bayes Classifier," in *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, Nanjing, 2015, pp. 638-642. DOI: 10.1109/PIC.2015.7489926
- [13] Trudgian D. C. , "Spam Classification Using Nearest Neighbour Techniques," in *Z. R. Yang, H. Yin, & R. M. Everson (Eds.), Intelligent*

- Data Engineering and Automated Learning -- IDEAL 2004: 5th International Conference*, Exeter, UK., August 25-27, 2004. Proceedings Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 578–585. DOI: https://doi.org/10.1007/978-3-540-28651-6_85
- [14] L. Firte, C. Lemnaru and R. Potolea, "Spam detection filter using KNN algorithm and resampling," in *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, Cluj-Napoca, 2010, pp. 27–33. DOI: 10.1109/ICCP.2010.5606466
- [15] Qiang Wang, Yi Guan, Xiaolong Wang, "SVM-Based Spam Filter with Active and Online Learning", School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China.
- [16] O. Amayri and N. Bouguila, "Online spam filtering using support vector machines," in *2009 IEEE Symposium on Computers and Communications*, Sousse, 2009, pp. 337–340. DOI: 10.1109/ISCC.2009.5202287
- [17] Kolesnikov O., Lee W., and Lipton R., "Filtering spam using search engines," in *Technical Report GITCC-04-15*, 2003, Georgia Tech, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, 2004–2005.
- [18] Shrivastava, J. N., and Maringanti, H. B., "E-mail spam filtering using adaptive genetic algorithm," in *International Journal of Intelligent Systems and Applications*, 2014, Vol. 6(2), pp. 54.
- [19] Y. Tan, G. Mi, Y. Zhu and C. Deng, "Artificial immune system based methods for spam filtering," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, Beijing, 2013, pp. 2484–2488. DOI: 10.1109/ISCAS.2013.6572383
- [20] Trudgian D. C., (2004). "Spam Classification Using Nearest Neighbour Techniques," in Z. R. Yang, H. Yin, & R. M. Everson (Eds.), *Intelligent Data Engineering and Automated Learning -- IDEAL 2004: 5th International Conference, Exeter, UK*, August 25–27, 2004. Proceedings Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 578–585. https://doi.org/10.1007/978-3-540-28651-6_85
- [21] Q. Luo, B. Liu, J. Yan and Z. He, "Design and Implement a Rule-Based Spam Filtering System Using Neural Network," in *2011 International Conference on Computational and Information Sciences, Chengdu, China*, 2011, pp. 398–401. DOI: 10.1109/ICCIS.2011.125
- [22] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, (). "A Bayesian approach to filtering junk e-mail," (PDF). In *AAAI'98 Workshop on Learning for Text Categorization*. 1998.
- [23] Graham P., "A plan for spam," (2002). Retrieved from: <http://www.paulgraham.com/spam.html>
- [24] Li K., Zhong Z., (2006) "Fast statistical spam filter by approximate classifications," in *SIGMETRICS Perform Eval Rev*, Vol. 34(1), pp. 347–358. ISSN 0163-5999
- [25] Shi L., Wang Q., Ma X., Weng M., Qiao H, "Spam email classification using decision tree ensemble," in *J. Comput. Inf. Syst.*, 2012, Vol. 8(3), pp. 949–956.
- [26] Bahgat E. M., Rady S., and Gad W., (2016). "An E-mail Filtering Approach Using Classification Techniques," In T. Gaber, A. E. Hassanien, N. El-Bendary, & N. Dey (Eds.), *The 1st International Conference on Advanced Intelligent System and Informatics (AISII2015)*, November 28–30, 2015, Beni Suef, Egypt Cham: Springer International Publishing. , pp. 321–331. DOI: https://doi.org/10.1007/978-3-319-26690-9_29