Web Service-enabled Spam Filtering with Naïve Bayes Classification

Wanqing You, Kai Qian, Dan Lo
Dept. of Computer Science
Southern Polytechnic State University
Marietta, GA, USA
{wyou, kqian, clo}@spsu.edu

Prabir Bhattacharya, Minzhe Guo
Dept. of Electrical Engineering and Computing Systems
University of Cincinnati
Cincinnati, OH, USA
{Bhattapr, guome}@ucmail.uc.edu

Ying Qian
Dept. of Computer Science
East China Normal University
Shanghai, China
yqian@ecnu.edu.cn

***Abstract*—Electronic mail has nowadays become a convenient and inexpensive way for communication regardless of the distance. However, an increasing volume of unsolicited emails is bringing down the productivity dramatically. There is a need for reliable anti-spam filters to separate such messages from legitimate ones. The Naïve Bayesian classifier is suggested as an effective engine to pick out spam emails. We have developed an anti-spam filter that employs this content-based classifier. This statistic-based classifier was trained on Eron Spam Dataset, a well-known spam/legitimate email dataset. We developed this filter as a Web Service, which would consume the emails user uploads and give back the predicted probability that in what degree the given email is spam. This engine was achieved by Resteasy technology, and consists three phases to train pre-labeled emails and then apply Naïve Bayes theorem to calculate email's spamicity.**

Keywords—Spam filter; Naïve Bayesian classifier; Web Services; Big data

I. Introduction

Electronic mail is an increasing popular way for communication. However, it is prone to misuse. Spam message is one of these cases, which is a blind posting of unsolicited emails to a very large number of recipients. In a research paper the statistics shows that spam emails have accounted for 68.8% of all email traffic in 2012[1]. Spam message is not only a waste of user's time; it can also spread harmful message and virus, which is more harmful and cost more. To avoid being detected, spam can be created in a variety of different styles and complexities. Some spam is just plain text sometimes with a URL; some is cluttered with images and/or attachment. By analyzing its content a spam typically would fall into the following several common categories: gambling, degrees/diploma, diet/weight loss, jobs/money mules, and phishing, scam and so on. These categories are rarely seen in legitimate message and can be helpful to differentiate legitimate message from spam message, which makes text-based classifiers to be used to filter out spam emails.

Our research is based on the text classifier. Although spam messages show up in different styles, the plain text based email is still the mainstream. The keyword patterns in spam messages should be manually crafted, and they also need to be tuned to each user to get a better result. With the aid of data mining, the process to separate spam emails from the ham emails can be done more efficiently. Text classification is one of the text mining technologies, and is the basis of our work. We applied the Bayesian classifier in our work, which is a supervised learning method that bases on machine learning. Bayesian classifier identifies spam e-mail by referring to the knowledge gained in training phase. In this prerequisite phase, emails have been manually classified as spam or non-spam and the features for distinguishing spam e-mails from normal e-mails are set up. Proposed by Sahami et al. [2], this content-based classifier has proved to be very effective.

A Naïve Bayes model is the most restrictive form of the feature dependence spectrum. It is a probabilistic classifier based on the assumption that typically tokens are words independent. Naïve Bayes classifier has been studied from 1950s, and still remains to be a popular method for text categorization with word frequencies as the features. Combining with appropriate preprocessing, the Naïve Bayes classifier can categorize text in an accurate manner, which can be shown in [3] where a comparison has made of the three algorithms, namely, the KNN algorithm, Naïve Bayes algorithm and reverse DB SCAN on Enron's spam/ham email dataset and they obtained good results from all algorithms.

Hadoop framework for big data processing is a concept used in our paper. To make this into an online system that can consume the user's input for training to update learned knowledge or classification, a big data-analyzing framework is needed. Especially, when the sample dataset contains a huge volume of emails. This platform is widely used in data mining. It helps to process a big dataset by splitting data into independent chunks and processing them in parallel. It can be integrated with popular databases to provide more flexible data storage system. There are researchers who focus on the development of the next generation of data storage system.

RESTful Web Service is another concept used in our work, which is a method of communication of different application in network. There are several implementation of JAX-RS, Resteasy is a JBoss project that provides various framework to help build RESTful Web Service that fully certified JAX-RS implementation. To make our system scalable and pluggable to other system, we make it a web service enabled system, which means user can call the spam detection function simply via a universal way --HTTP protocol.

In our work, we combine the techniques mentioned above to do spam detecting. Based on the famous e-mail dataset, we utilize several useful framework to build a novel system for spam detection. Bayesian text classifier is the essential concept of our work. Hadoop, as a big data processing framework, is employed to make the training phase of Naïve Bayes more efficient, and Web Service is applied to provide universal resources identifiers to make the system scalable and pluggable.

The most significant contributions of our work include the utilization of Hadoop big data analysis framework and the concept of Web Services. With these two technologies, our system is efficient and scalable.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to the related works. In Section 3, we introduce the experimentation used in our work. The results and description are presented in Section 4. Finally, in Section 5, we conclude and discuss future work.

II. Related Works

Email has become a main method of mail services nowadays, people use emails for business purposes or personal communication. As research statistics shows [4], nearly 144.8 billion emails were sent every day.

Spam email issue has become one of the hottest Internet issues. Currently, researchers have already proposed a number of methods to deal with spam detection based on machine learning algorithms. Among them, Naïve Bayes classifier is suggested as a more effective method, which is a text-based classifier.

The first spam email filter using Naïve Bayes classifier is Jason Rennie's ifile program. Since then, it has attracted considerable attention by researchers. Sahami et al. [2] published the first Bayesian spam-filtering algorithm. Graham [5] proposed more filtering rules by greatly reducing the false positive rate to make a better spam filtering.

Mehran et al [6] examined methods to construct filters automatically to eliminate unsolicited emails from a user's mail stream. They cast this problem in a decision-theoretic framework and made use of probabilistic learning methods. Based on the concept of text classification, they considered domain-specific features in addition to the raw text of mails. Their method was proved to be accurate and mature enough for deployment.

Spam emails can be presented in different styles and complexities. To avoid detection, spammers create their spam in more and more complicated styles. Embedded advertisement, images, and attachments are some of the common ways to make up spam emails, rather than just the plain text. There are different algorithms for filtering spam messages in different styles. For filtering image spam mails are algorithms such as the NDD, SIFT, TR-FILTER, and also various algorithms for filtering text spam emails like SVM, KNN. Ketari et al.'s work [7] explored the major image spam filtering technologies. Deshmukh et al. [8] gave a survey of various kinds of algorithms for filtering text-based and image-based emails.

Ho [9] proposed a sim-hash algorithm to detect spam email, which combines fingerprinting technique and big data processing. This is a similarity-based method, so that a large dataset is needed to extract out similarity among spam emails. Also, Hadoop Map/Reduce framework is used for data processing and a big enough storage system is needed to store the large number of fingerprints.

Our work is based on the Naïve Bayes theorem and some of the works stated above. The advantages of Bayesian classification are: (1) Efficiency: Naïve Bayesian algorithm scans all the words in samples only once to get the occurrences of each token in mails. (2) Self-evolution: Naïve Bayesian spam filtering system can continue receive user input for training to update learned knowledge. (3) Storage: by combining with Hadoop framework, the words of messages are stored in HDFS, sometime other external databases. (4) Accuracy: Naïve Bayesian classifier not only counts the appearance of tokens in spam emails but also in legitimate emails, and by allocated different weights. In addition, we developed the Naïve Bayes classifier into a Web Service-enabled system to make it more scalable and adaptable.

III. Design

Our experiment contains two phases: training and classification. In the training phase, we employed Map/Reduce in Hadoop technologies for Big Data analysis to count the occurrences of tokens. The dataset used is a known corpus, by counting the appearance of each token in the dataset the features or knowledge is learned. Based on the occurrence, each token is assigned a probability to represent its contribution in determining an email to be spam. This knowledge of tokens is used in the classification phase to predict the probability of new incoming emails to be spam. If a new email is confirmed to be a spam or a ham, all its tokens are recorded in the corresponding vocabulary list to update the knowledge obtained before. This self-learning function would make the prediction more accurate.

A. Dataset

We used Enron's dataset [10] for training and testing. The total number of training emails is 6000, consisting of 1500 legitimate emails and 4500 spam emails, which are already labeled as spam or ham manually and no encoding in email content. After training, we used a testing dataset, which is made up of 330 spam emails and 270 emails to test our system.

Messages in the sample dataset have a format shown in Fig.1.

Subject: your prescription is ready . . oxwq s f e
low cost prescription medications
soma , ultram , adipex , vicodin many more
prescribed online and shipped
overnight to your door ! !
one of our us licensed physicians will write an
fda-approved prescription for you and ship your
order overnight via a us licensed pharmacy direct
to your doorstep . . . . fast and secure ! !
click here !
no thanks , please take me off your list
ogrg z
lqlokeolnq
lnu

Fig.1 Email content sample

B. Preprocessing

Naïve Bayesian classifier is a supervised method to classify spam emails; in the training phase we need to count the occurrence of every token in the corpus and calculate the probability that message is spam based on the each token's presence. Because of the large volume of tokens, we applied Hadoop's Map/Reduce framework for Big Data to deal with the dataset to extract features from existing emails. In order to improve the performance, we also consider the following pre-processing techniques.

- Find out links or URLs in emails and transform them to a simple word and collect them into a black list if the links appear in spam emails. Combining with the black list, the URLs can become special flags to determine if a message is spam.
- Use Stanford's open API [11] for English words' lemmatization. For example, "gain", "gaining" and "gained" should be regarded as a same feature "gain" in vocabulary. In this way, the size of features can be limited in some degree, which can reduce the processing time.
- Another thing to do is to exclude some useless or not so important words from the feature list. Words, like prepositions, contribute little to determine if an email is spam or not. Thus, we exclude them from the feature list to make the prediction more accurate as well as to improve the efficiency.
- Filter out numbers but keep the special signs like "$", which is a very common feature in spam emails, and filter out words that have length less than three that may have not practical meaning and may contribute less for the determination of a message to be a spam or not.
- Hadoop Map/Reduce framework for word count. This is a software framework for processing vast amounts of data in parallel. A Map/Reduce job will consume the input in any format and the input data is split into independent chunks that are processed by *map* tasks in parallel. Then the framework emits key-value pairs and sorts the outputs of the maps, which are inputted to the *reduce* tasks [12]. In general, both the input and output are stored in HDFS, Hadoop file system.

C. Training

The Naïve Bayesian classifier stated above is employed to get the posterior probabilities of each token. Naïve Bayes classifier is based on the assumption that every token is independent so that the presence or absence of a particular word would not affect the other words' contribution to the spamicity of an email.

The general processing of Naïve Bayes classifier can be described as follows: get a labeled sample and train it to build up the probabilities of each token in corpus, then the word probability obtained in the previous step would be used to compute the probability that an email with a particular set of words in it belongs to either category. The procedure is shown schematically in Fig.2.

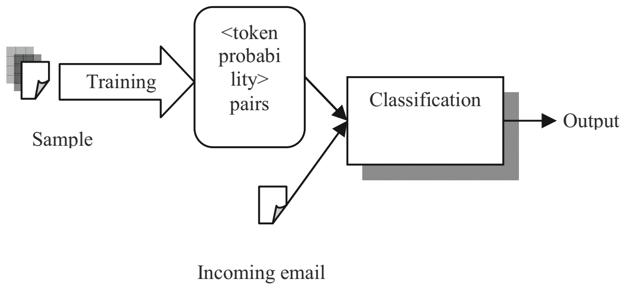


Fig.2 General processing flow

In training phase, we need to count the bag of words in the sample dataset, which has been labeled as spam or ham beforehand manually. In order to process the huge dataset efficiently, Hadoop's Map/Reduce framework is employed here. The sample dataset is first being uploaded to HDFS, then split into independent chunks being processed by different *map* tasks, which would emit <key,value> pairs as output. The output here becomes the input of *reduce* task and the result would be stored in HDFS, which is shown in Fig.3.

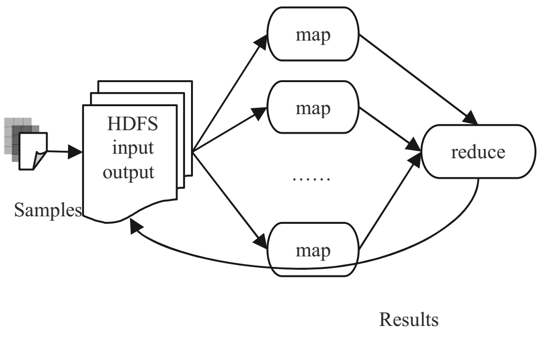
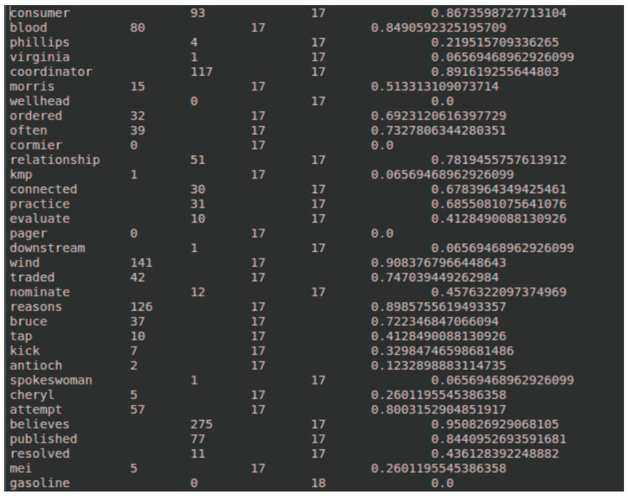


Fig.3 Map/Reduce processing flow

Part of the training result is shown as Fig.4. There are four columns: tokens, the number of appearance in spam messages, the number of occurrence in ham message, and the spamicity of the token.



```
consumer                93              17                      0.8673598727713104
blood           80              17              0.8490592325195709
phillips                4               17                      0.219515709336265
virginia                1               17                      0.06569468962926099
coordinator             117             17                      0.891619255644803
morris          15              17              0.513313109073714
wellhead                0               17                      0.0
ordered         32              17              0.6923120616397729
often           39              17              0.7327806344280351
cormier         0               17              0.0
relationship            51              17                      0.7819455757613912
kmp             1               17              0.06569468962926099
connected               30              17                      0.6783964349425461
practice                31              17                      0.6855081075641076
evaluate                10              17                      0.4128490088130926
pager           0               17              0.0
downstream              1               17                      0.06569468962926099
wind            141             17              0.9083767966448643
traded          42              17              0.747039449262984
nominate                12              17                      0.4576322097374969
reasons         126             17              0.8985755619493357
bruce           37              17              0.722346847066094
tap             10              17              0.4128490088130926
kick            7               17              0.32984746598681486
antioch         2               17              0.1232898883114735
spokeswoman             1               17                      0.06569468962926099
cheryl          5               17              0.2601195545386358
attempt         57              17              0.8003152904851917
believes                275             17                      0.950826929068105
published               77              17                      0.8440952693591681
resolved                11              17                      0.436128392248882
mei             5               17              0.2601195545386358
gasoline                0               18                      0.0
```

Fig. 4 Training result

D. *Classification*

The word count result was also stored in HDFS. The following part depicted how to apply the Naïve Bayes theorem to calculate the probability that an email is spam as the presence of a particular word.

First, we calculate the spamicity of a word as in

$$P(S/W) = P(W/S)*P(S) / (P(W/S)*P(S)+P(W/H)*P(H)) \quad (1)$$

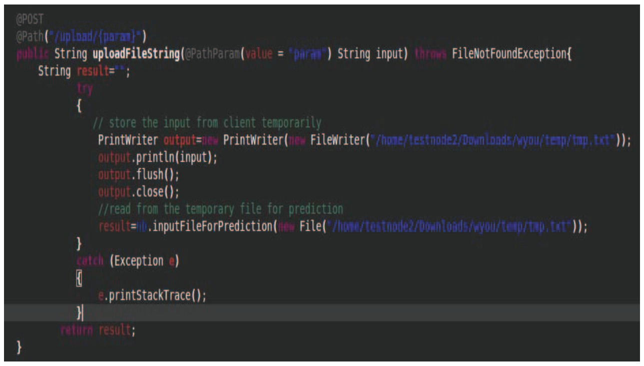
where P(S/W) denotes the probability that a message is spam with the word W in it; P(S) is the overall probability that any given message is spam; P(W/S) is the probability that a particular word appears in spam messages; P(H) is the overall probability that any given message is ham; and P(W/H) is the probability that a particular word appears in ham messages.

Most Bayesian spam filtering algorithms are based on formulas that hold strictly only if the words present in the message are independent of each other, which is not always satisfied (for example, in natural languages like English the probability of finding an adjective is affected by the probability of having a noun), but it is a useful idealization, especially since the statistical correlations between the individual words are usually not known. On this basis, one can apply the Bayes' theorem to calculate the probability that a message is a spam with the bag of words in it.

$$P = P_1P_2 \ldots P_N / P_1P_2 \ldots P_N + (1-P_1)(1-P_2)\ldots(1-P_N) \quad (2)$$

where, P_1, P_2 … P_N are the probabilities that a message is spam-knowing words P_1, P_2 … P_N in it respectively.

In our work, we have implemented a Web Service-enabled module to achieve the classification by applying the Naïve Bayesian classifier. In this module, the probability of each token is prepared for the classification for an incoming email. The reason why we make it a web service is that in this way, the spam-filtering module can be easily embedded or made use of other systems. A web service provides a universal interface, and can be scalable and adaptable to other systems. In this paper, we employed *Resteasy*, developed by JBoss, to implement this module. A simple *POST* method provided by this module to calculate the probability that a message is spam with the words in it is shown in Fig.5.



```
@POST
@Path("/upload/{param}")
public String uploadFileString(@PathParam(value = "param") String input) throws FileNotFoundException{
    String result="";
        try
        {
          // store the input from client temporarily
           PrintWriter output=new PrintWriter(new FileWriter("/home/testnode2/Downloads/wyou/temp/tmp.txt"));
           output.println(input);
           output.flush();
           output.close();
           //read from the temporary file for prediction
           result=nb.inputFileForPrediction(new File("/home/testnode2/Downloads/wyou/temp/tmp.txt"));
        }
        catch (Exception e)
        {
           e.printStackTrace();
        }
      return result;
}
```

Fig.5 Web Service method

In order to test the web service-enabled module for spam filtering, we developed a simple Java Web Application. Since the web service has reusable application components, it is able to interact with existing software. You can have other client applications to consume the services provided by this system. In Fig.6, you can have the client side either in Java or C# to access the resources in server.

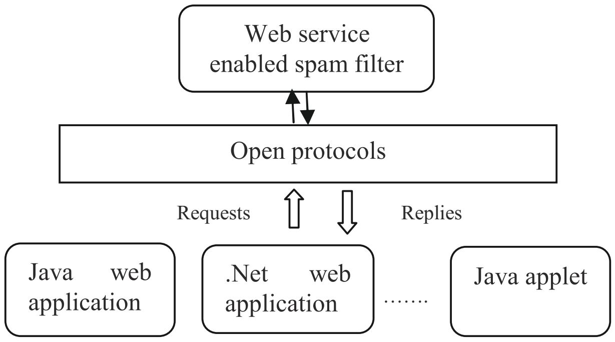


Fig.6 Web service enabled spam filter

E. *System Architecture*

The proposed system takes the sample dataset as the input. Sample emails are stored in HDFS and ready for training. Input is pipelined into a pre-processing block. In this block, the aforementioned pre-processing procedures are taken one by one. Then the output from the pre-processing block sent as the input to the Map/Reduce word count framework. The feature list extracted from the Map/Reduce framework is the basis for the Bayesian filter training to get the knowledge for classification. Fig. 7 shows the system architecture.

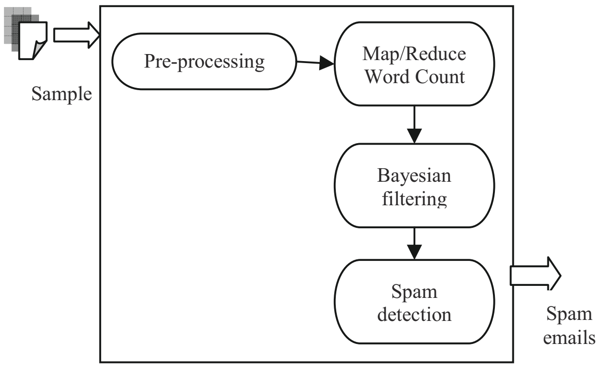


Fig7. System architecture

F. *Algorithm*

The algorithm is depicted in the following way. Once the training phase is over, the knowledge is learned and a feature list (with the tokens and the probabilities that they contribute to make a message spam) is ready for the classification of the incoming emails. According to formulation (2), the incoming emails are examined word by word - if the word is already in the feature list, we get its value and move forward. If the word is not in the feature list, we assign a probability value to it; in our work, we have set it as 0.5. By applying the Naïve Bayes theory, the probability that a message is spam given the bag of words in it is obtained. If the result is wrong after a checking by the user, the email would be sent back to training phase to update the feature list.

```
Populate excluded list EL, black list BL, and feature list FL
Initialize d=1.0, de=1.0
For all $p_i$ in doc d:
    If $p_i$ in FL
        d *= FL.getProbability($p_i$)
        de *= (1- FL.getProbability($p_i$))
    else
        d *= 0.5
        de *=0.5
Return (d / (d + de))
```

Fig.8 Algorithm

IV. Evaluation

In spam filtering, a false positive would always take place, which means to classify mistakenly a legitimate email as a spam. A false negative, which classifies a spam email as a ham, could also exist in spam filtering. A false positive could be more serious than a false negative because, in this case, the users would delete legitimate emails due to spam filters without reading them. Under such circumstances, it is acceptable to have some false negatives. Thus, it is important to keep the false positives up to a minimum value. However, according to [13], all techniques of the Bayesian approach allow some number of false positive and it cannot be eliminated completely. And they evaluated different threshold values to adjust different number of false positive and false negative to optimized anti-spam filter configuration. They also suggested the use of other techniques, like lemmatizer, blacklist and rule-based methods to optimize anti-spam filters.

Fortunately, there are some techniques that can be used to improve the performance. One that we have applied in our work is to use Stanford's APIs to do the word lemmatization. In our work, we do not consider the non-textual features in emails, like images and attachments, although there is a high probability that a message is a spam with only images or attachments. In the following part, the experimentation and the results are described in detail. Table I shows the environment setup for the experiment.

TABLE I. ENVIRONMENT SETUP

	Name	*Version*
Operating System	Ubuntu	14.04
Java	Java SDK	7.0
IDE	Eclipse	For Linux
Big Data Analysis Framework	Hadoop	2.3.0
Web Services Framework	Resteasy	N/A

The evaluation result of our work with a pre-processing phase is shown in table II, and the result without pre-processing is shown in table III. Here, 600 testing emails, with 330 spam emails and 270 ham emails are included in the testing dataset.

With the pre-processing, some insignificant tokens are excluded to improve the efficiency as well as the accuracy that is shown in the following tables. With the pre-processing, the false positive value was reduced about 31.6% while there is only a 1.3% false negative increase.

TABLE II. EVALUATION RESULT

	Result	
	Result of spam testing data	*Result of ham testing data*
Total	330	270
Classified as spam	308	39
Precision	93.33%	86.56%
False positive	N/A	14.44%
False negative	6.67%	N/A

TABLE III. EVALUATION RESULT

	Result	
	Result of spam testing data	***Result of ham testing data***
Total	330	270
Classified as spam	312	57
Precision	94.55%	79.89%
False positive	N/A	21.11%
False negative	5.45%	N/A

V. CONCLUSION AND FUTURE WORK

In this paper, we applied the Naïve Bayesian theory for spam filtering. We integrated Hadoop Map/Reduce framework for big data to process the large volume of sample emails. Also, we added a pre-processing phase while training, which will kick out some insignificant words, extract some typical features, and help improve the accuracy of email classification. With the training result, we achieved a moderate prediction when encountering a new incoming email. Moreover, we made it a Web Service, which is scalable and adaptable to other systems. This web services enabled system can consume requests from various clients regardless of their platform, which means it is not limited to a particular platform. Due to this mechanism, it can act as a standalone server to provide services for clients as well as play the role to be integrated into other systems.

As a future work, our system can be improved still further. For one thing, the current version of the system cannot deal well with malformed URLs. In future, we plan to make the training phase to be an online procedure, which means it will take in a collection of emails that have been already labeled and update the corpus database accordingly. Another improvement is to make the system learn from its fault. This means that if the system gives a wrong prediction for the incoming email, this email will be labeled manually and become part of the training dataset to be trained again.

There are some traits that can help to improve the Naïve Bayes' classification accuracy, which we also plan to add into our system. One of these traits is the mail header. Features like blank "From" field, list a lot of addresses in the "To" field, have too many digits in email address and the use of dark red colors, all capital characters in subject are very common in spam messages. In future, we plan to build a comprehensive checking module for mail header.

REFERENCES

[1] http://royal.pingdom.com /20 13 /0 I 116 /internet- 2012-in -numbers

[2] M. Sahami, S. Dumais, D. Heckerman., and E. Horvitz,, "A Bayesian approach to filtering junk email," Proc. AAAI Workshop on Learning for Text Categorization, 1998, AAAI Technical Report WS-98-05.

[3] Harisinghaney, A., et al. "Text and image based spam email classification using KNN, Naïve Bayes and Reverse DBSCAN algorithm." IEEE Conf. Optimization, Reliabilty, and Information Technology, 2014.

[4] https://www.linkedin.com/pulse/article/20140405173711-113496637-to-spam-or-not-to-spam-part-1-open-the-tin

[5] P. Graham (2003), Better Bayesian filtering.

[6] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, "A Bayesian Approach to Filtering Junk Email"

[7] L.M. Ketari, L..M. Chandra, and M. A. Khanum. "A Study of Image Spam Filtering Techniques," 4th IEEE Internat. Conf. Computational Intelligence and Communication Networks, ,2012.

[8] S.S. Deshmukh., P.R. Chandre, "Survey on: Naive Bayesian and AOCR Based Image and Text Spam Mail Filtering System", International Journal of Emerging Technoogy and Advanced Enginerring

[9] P. T. Ho, H. S. Kim, S. R. Kim, "Application of sim-hash algorithm and big data analysis in spam email detection system, " Proc. 2014 Conference on Research in Adaptive and Convergent Systems, pp. 242-246

[10] http://www.aueb.gr/users/ion/data/enron-spam/

[11] http://nlp.stanford.edu/software/corenlp.shtml

[12] http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[13] V. P. Deshpande, R. F. Erbacher and C. Harris, "An evaluation of Naïve Bayesian anti-spam filtering techniques." Proc. of the 2007 IEEE Workshop on Information Assurance.