Open Access

Open Journal of Big Data (OJBD)
Volume 1, Issue 1, 2015

http://www.ronpub.com/journals/ojbd
ISSN 2365-029X

Cognitive Spam Recognition Using Hadoop and Multicast-Update

Mukund. Y. R [A], Sunil Sandeep Nayak [A], K. Chandrasekaran [B]

[A] National Institute of Technology Karnataka, Surathkal, Mangalore 575025, India, {mukund.mukund16, sunil.nayak133}@gmail.com
[B] National Institute of Technology Karnataka, Surathkal, Mangalore 575025, India, kchnitk@ieee.org

Abstract

In today's world of exponentially growing technology, spam is a very common issue faced by users on the internet. Spam not only hinders the performance of a network, but it also wastes space and time, and causes general irritation and presents a multitude of dangers - of viruses, malware, spyware and consequent system failure, identity theft, and other cyber criminal activity. In this context, cognition provides us with a method to help improve the performance of the distributed system. It enables the system to learn what it is supposed to do for different input types as different classifications are made over time and this learning helps it increase its accuracy as time passes. Each system on its own can only do so much learning, because of the limited sample set of inputs that it gets to process. However, in a network, we can make sure that every system knows the different kinds of inputs available and learns what it is supposed to do with a better success rate. Thus, distribution and combination of this cognition across different components of the network leads to an overall improvement in the performance of the system. In this paper, we describe a method to make machines cognitively label spam using Machine Learning and the Naive Bayesian approach. We also present two possible methods of implementation - using a MapReduce Framework (hadoop), and also using messages coupled with a multicast-send based network - with their own subtypes, and the pros and cons of each. We finally present a comparative analysis of the two main methods and provide a basic idea about the usefulness of the two in various different scenarios.

Type of Paper and Keywords

Regular research paper: *Spam, Hadoop, Naive Bayes, Multicast, Machine Learning, Cognition, Distributed Systems*

1 Introduction

A system in a network might receive numerous messages which do not support any useful cause. A lot of them are advertisements, and a few of them malicious in nature. This unsolicited message is termed as *Spam*. These Spam messages cause the wastage of a system user's time and also the resources. E-Mail is one of the common media where the spamming is considered to be a serious problem. The spam mails not only contain advertisements which may be considered harmless but also play part in malicious activities like phishing, spreading Trojan viruses etc,. The amount of spam mail grew exponentially over the following years, and today composes some 80 to 85 percent of all the e-mail in the World, by a "conservative estimate" [1].

It would be a great benefit to the system user if the system were able to recognize the spam mail before the user opened it. This ability is called *spam recognition*. Spam recognition is a technique which uses machine learning concepts to train the system to recognize the mail as

spam with reference to its content. Spam recognition can be applied either on a user machine or on the mail server itself. These programs which implement spam recognition are called spam filters [2]. They can be applied either on a user machine or on the mail server itself [3]. The most commonly used filters are Bayesian filters [4], which use the Bayes theorem to classify the mail as spam with reference to a probability model constructed on a known dataset of spam mails [5].

When a system classifies a mail as spam and if that knowledge gained by that particular system is shared among the other systems in the network the collective spam filtering efficiency increases. This method of collective learning between the systems is called cognition [6].

The spam mails today are being modified so that the existing filters do not recognize them as spam. The spam filtering system needs to adapt to this change in pattern of the spam mail. This can be achieved through the concept of collective learning. This paper gives the idea of collective learning employed to recognize spam. In doing so it discusses the overheads encountered and also a new technique which can be used to overcome that overhead.

In this paper, we make the following research contributions:

1. *Collective learning using Multicast-Update:* In the field of spam filtering, a huge part of the work has been done to improve the existing algorithms used. Much importance has not been given to the underlying infrastructure. Even though we do not deny that the work on the algorithms is important, the paper concentrates on improving or using the infrastructure and thereby influencing the accuracy and the robustness of the entire system. An algorithm is given which uses the simple concept of multicasting to share the knowledge on spam between the systems that form a network. This sharing of knowledge increases the richness of the learning in the individual systems, which makes the systems more intelligent resulting in a better performance.

2. *Collective learning using the Hadoop MapReduce framework:* Distributed systems have many favourable characteristics like scalability, resilience etc. [7]. The paper also talks about the concept of a spam filtering system employing a distributed architecture. An algorithm is given where the MapReduce framework is used along with a classification technique to gve us a spam filtering system. This method conquers certain disadvantages of the earlier proposed Multicast-Update method, like having to perform the learning seperately for each system, the network overhead involved in knowledge sharing etc. Thus a spam filtering system is proposed which has an improved performance and also the favourable characteristics of a distributed system.

3. *Increasing the efficiency of the MapReduce framework for learning problems using a flag:* In the proposed distributed method of spam filtering, we see that the system has to perform a $reduce()$ everytime a false positive occurs. So to overcome this overhead a $newreduce()$ function is proposed which employs a simple technique of checking a flag before carrying out any $reduce()$ operation. This method decreases the overhead by a large amount. The technique of using a flag can be employed for any application using a MapReduce framework where the data in the file systems change very often i,e. where $reduce()$ operations have to be performed very often.

Even though all these methods are given for the application of spam filtering. They can be used for any learning problem, which has to deal with a large amount of data and where the data is subject to change very often.

The structure of the paper is as follows, section 2 describes the Naive-bayesian Classifier model, section 3 describes some of the related work, section 4 gives the algorithm which uses Multicast-Update, section 5 gives the algorithm which uses the MapReduce framework, followed by section 6 which gives a mathematical analysis on the runtime performance and the overheads that occur. section 7 gives the results of the simulations conducted thus gives an analysis of the classifier algorithm and also proposed methods for the Mapreduce framework and finally section 8 concludes the paper in accordance with the results obtained.

2 NAIVE-BAYESIAN CLASSIFIER

Spam recognition in most cases involves the Naive-Bayesian classifier. This classifier uses the factors of the text such as word count and associated features to train or build a probabilistic model for each occurrence of a word. The incoming mail is weighted based on this probabilistic model and the final weight calculated determines if the mail is spam or not. For instance let us consider the word ”sales” which occurs in a lot of spam mails. The Bayesian classifier is trained such that it assigns a very high probability of the word being associated with a spam mail to the word ”sales”. This decreases the probability of the word being associated with a normal mail. When the classifier encounters this word it assigns the weight accordingly and hence is classified as spam.

The formula by equation 1 gives the probability func-

tion of the word being associated with a spam mail.

$$P(S|W) = \frac{P(W|S) \cdot P(S)}{P(W|S) \cdot P(S) + P(W|NS) \cdot P(NS)} \quad (1)$$

where:

- $P(S|W)$ is the probability that the given mail is a spam, knowing the word "sales" is in it.
- $P(S)$ is the Total probability that any mail is spam.
- $P(W|S)$ is the probability that the word "sales" appears in spam messages.
- $P(NS)$ is the overall probability that any mail is not spam.
- $P(W|NS)$ is the probability that the word "sales" appears in general mails.

3 Related Work

Wang et al. in [8] applies the concept of spam recognition to the popular social networking site Twitter. Based on the analysis of the data,he defines the *follower* and the *friend* relationships among the users. There are two approaches that are discussed a graph based feature and content based feature approach, In the graph based features of the tweet the *reputation* based feature which is proposed here has the best performance of detecting abnormal behaviour. Applying Various machine learning algorithms the conclusion is that the Bayesian classifier has the best performance with respect to the *F* factor.

The paper by Damiani et al. [9] speaks about a P2P system where spam is collaboratively detected and filtered. They use a protocol and the hierarchy of user level, peer level and super-peer level to deal with the spam detection. The paper concludes on the advantages of the distributed architecture where there is no centralized authority of control. Thus showing us that the P2P mechanism generally applied as sharing technique can be used to address an important problem like spam recognition.

Piskorski et al. in [11] explores the possibility of using the linguistic features in a mail for the task of spam recognition. Dai et al. in [10] give us a Trusted Execution Emvironment for a cloud computing environment. The proposed TEE does not have any extra overhead regarding the performance of the Virtual Machines used. It can suppport a wide variety of application security needs from cryptographic libraries to trustworthy software. This type of a secure environment in cloud which ensures that there is no input given to the systems from an untrusted system is very desirable in implenting our method because it is assumed in our method that all the systems are trustworthy and hence the exchange of spam knowledge takes place without any security concerns.

Yang et al. in [12] also talk about the Naive Bayes approach extensively. The paper deals with the various sub-approaches and the intricacies of Naive Bayesian methods. A model which is based on the classifier error weights is proposed and with experimentation it is shown that the method is effective. Sasaki et al. in [13] gives us a new spam detection technique based on text clustering on a vector space model. It applies a spherical k-means algorithm on all the mails and generates centroid vectors which are further labelled as spam or non-spam based on the number of spam and non-spam mails in those particular clusters. A mail is classified based on the cosine similarity between the mail vector and the centroid vectors of all the clusters that are calculated and based on this the mail is classified into the most similar cluster. Chang in [14] discusses Buisness Intelligence as a Service, on cloud. The discussion is about providing an integrated service consisting of the Heston Volatility and Pricing as a Service (HVPaaS) and the Business Analytics as a Service (BAaaS).

Tan et al. in [15] give a collaborative intrusion detection framework for Big data security. To improve the accuracy and performance for the IDS the framework should be able to access the network data of all the individual systems on the network. They use the MapReduce framework, for the network data summarization. Chang et al. in [16] propose a framework to achieve Big data security. They provide a multi-layer security which is much more effective than the current systems which give a single solution. It is shown that this framework could isolate and quarantine 97.53% of trojans and viruses. Garg et al. in [17] talks about how in previous applications of collaborative spam filtering users shared the email fingerprint of the mails that are known to be spam, a new method is proposed where-in the spam filters of the users are shared as an alternative method for spam recogniton applications. This new approach has an advantage of reduced communication among the users.

Kakade et al. in [18] gives us an idea of an SVM classifier using a MapReduce framework. Priyadarshini et al. in [19] talks about a Document based CMS on a Hadoop File system. As given in [17] the concept of collaborative spam filters is interesting, but the underlying architecture for that purpose is not very common and easily available and thus not desirable for a simple spam filtering system. The method of using clustering algorithms as given in [13] takes a large time for clustering, even though effective the time complexity of the method is undesirable. P2P networks can be applied to networks consisting of a large number of systems but not for a small network. As stressed in [8], [12] the use of the naive-bayesian classifier for the classification problem is considered in this

paper.

The concept of a collaborative IDS given in [15] is a similar problem compared to the collective spam filtering, but the method requires a lot of additional infrastructure for that purpose. It is desirable to develop a system which does not require extra infrastructure and also uses the concept of collective learning. The framework proposed in [16] by Chang et al. is very effective in protecting a certain cloud application against attacks and can be employed to prevent mails containing viruses and trojans.

However the spam mails which are more of an inconvenience rather than a security concern should also be blocked by the spam filtering system. Considering all the existing solutions to spam filtering, we propose a unique method which gives the users a more desirable solution. The kind of a framework given in [14] can make use of collective spam filtering. Since the systems involved, have a smilarity in preferences regarding spam and ham and also since it is a trusted network, collective spam filtering can be employed to save time of the employees which is otherwise spent in clearing spam mails. As the number of employees increase the time saved increases and so does the impact on the company's productivity.

Table 1 gives the various solutions for the problem of spam filtering. However most of these techniques can be overcome by the spammers, if they can adapt their sending methods. To develop a system which can recognize new patterns, the method of using Machine Learning algorithms is the best approach. The advent of systems having high computing power and classifiers which are computationally less expensive, like the naive-Bayesian Classifier makes the implementation of such systems easier.

4 Algorithm Using Multicast-Update

Multicasting is when a system in a network, sends a single message to a group of other systems in the network [20]. This method is more efficient than multiple unicasting where the same message is duplicated and sent separately to each of the system. In multicasting the intermediate routers which are connected to the systems involved in the group do the duplication and forward the duplicates to the group. In this paper we present multicasting as one of the methods employed to update the probability model of the systems in case of a wrongful classification by the existing probability model. Whenever the model classifies an incoming spam message as not spam the system asks the user for his/her opinion of whether it is a spam or not, if the user's opinion is different compared to the classification that has been made by the model then the probability models of all the systems are updated.

Each of the systems have the same algorithm and initially all the systems share their training sets among each other. Each instance of the mail data is passed to a function $gen\_word\_vector()$ which parses the key words in the mail and builds a word vector [21] containing the key words, the number of occurrences of that word in the mail and also contains the information on the nature of the mail(spam or not spam). A probabilistic model $probability\_model()$ is built using these word vectors which contains all the words and their associated probability weight of them occurring in a spam.

After this build the system is ready to classify the mail and during the learning phase whenever a mail is classified it inquires the user to again verify if its classification is accurate. If the user disagrees with the classification then the model needs to be updated so the system updates its model and also sends the update data to its peers with the function $multicast\_send\_newresult()$ which sends the mail data along with correct classification. The systems which receive the mails also update their models thereby employing cognition or collective learning among the peers.

5 Algorithm Using The MapReduce Framework

Another way to implement the Naive-Bayes classifier discussed above would be to use the distributed file system - Hadoop [24]. We would be using the Hadoop API extensively for doing so, especially the map and reduce functions [22]. However, using the normal $reduce()$ function would lead to an additional overhead (which would lead to a proportional increase in runtime [25]).

To overcome this our method would involve writing the value of a particular flag to a file. The function of this flag would be to check if the probability model has changed. We would need to know this because we would be classifying each new mail instance using the probability model we have and this model must be the most updated version of itself, else there would be no point of the collective learning mechanism we have developed. The suggestion is to make slight changes to the $reduce()$ [22] mechanism and this changed function is addressed as "$new\_reduce()$".

5.1 Training

The process of training involves, creating the initial classifier model from the available training datasets. The classifier is built on the knowledge that is extracted from each of the examples in the training set which, are marked *spam* or *not spam*. Based on these examples, the classifier learns, as to when a mail should be classified

Table 1: Description of the current Spam Recognition techniques

Method used	Description	Advantages	Disadvatages	Examples
Blacklisting of IP	The sender's IP is black-listed referencing it against an available blacklist	A fast method of spam classification	May Result in False Positives for shared IPs	DNS Black Listing [27]
Authenticating Senders	The SMTP servers is checked on the list of authorized servers	The identity of the sender is verified by the SMTP server and backscatter of spam is reduced	Forwarded E-Mails may get through	SenderID [28]
Grey Listing	E-mails from unfamiliar sources are blocked. Assumes valid SMTP servers retries sending the mail while a spamming server wont.	User interaction is reduced and the amount of system resources used is also reduced.	Sending methods can be easily changed to adapt and there is a delay in the E-Mail delivery.	[29]
Challenge Response	System stores the mails received from unknown senders and sends back a challenge to verify if the sender is human or a bulk sender.	One time only process	Address spoofing can be done, also may result in spam backscatter.	Tagged message delivery agent [30]
Collaborative Spam Filtering	Mail fingerprints are compared against a shared list from trusted systems.	Uses the collective intelligence and since the classification is human, accuracy is high	Highly dependent on proper training	Distributed Checksum Clearninghouse [31], [17], [32]
Heuristic based	Compares mail against pre-defined rules to indentify traits of spam.	Training Overhead does not exist.	System is not adaptable to new patterns.	SpamAssasin [34]
Machine Learning	Knowledge is derived from a supplied Training set, this knowledge is then used to classify mail.	System is adaptable to new patterns in spam.	Needs good training datasets to derive knowledge.	Bayesian Spam filtering [4]

as 'spam'. In our algorithm the initial training dataset is shared among all the systems in the network which helps in collective learning.

Initially there exists n systems in the network each with its own training data. We would need to create a file, "mapfile" in each system. This system would contain a $dirty - flag$ (which will be explained further) and the output of the $map()$ [22] function when it is run, i.e, it would contain a map (key, value pair [22]) of all the words available in the training set.

Next, we would have to run the $reduce()$ [22] function for every key present in the mapfile (to count the number of occurrences of each word) and store the collective output in a dictionary say "r". We would use "r" to generate the probability model "pt" using the Naive-Bayesian classifier (parse r's keys and calculate the probability at every step). We would have to send the dictionary "r" to all the systems in the network which helps in sharing the knowledge of spam. The other systems use this dictionary and build the probability model. Since all the system share their dictionaries, the initial probability model is the same for all the systems. This concludes the initial training phase of the implementation.

5.2 Working

Post completion of the training phase, each system would have to wait for mails/messages from which we will have to segregate spam. Until such a mail is received, the system keeps doing its own job or remains waiting for the mail (we place no restriction on this, it can be synchronous or asynchronous). Once a mail is received, the actual usage of the training done starts.

Firstly, the system checks its mapfile to see if the $dirty - flag$ is set to 1. If it isn't, the system classifies the mail based on the probability model "pt" as spam or not-spam. It then asks the user whether the classification is right. If the user agrees with this classification, no step is taken as the training holds. Else, if the user finds a discrepancy, the word-vector (the map, as it was called before), "m" changes. This change has to be written to the mapfile of the corresponding system. Note that the maps in the mapfile of the other systems need not be changed by virtue of how Hadoop works. Then, for every key in m, the function "$new\_reduce()$" is called. This function works in the same way as the $reduce()$ function except for the fact that it sets the $dirty - flag$ to 1 in mapfiles of all the systems it goes through.

This would reduce the overhead required to open and write to the file every time we called $reduc()e$ for every

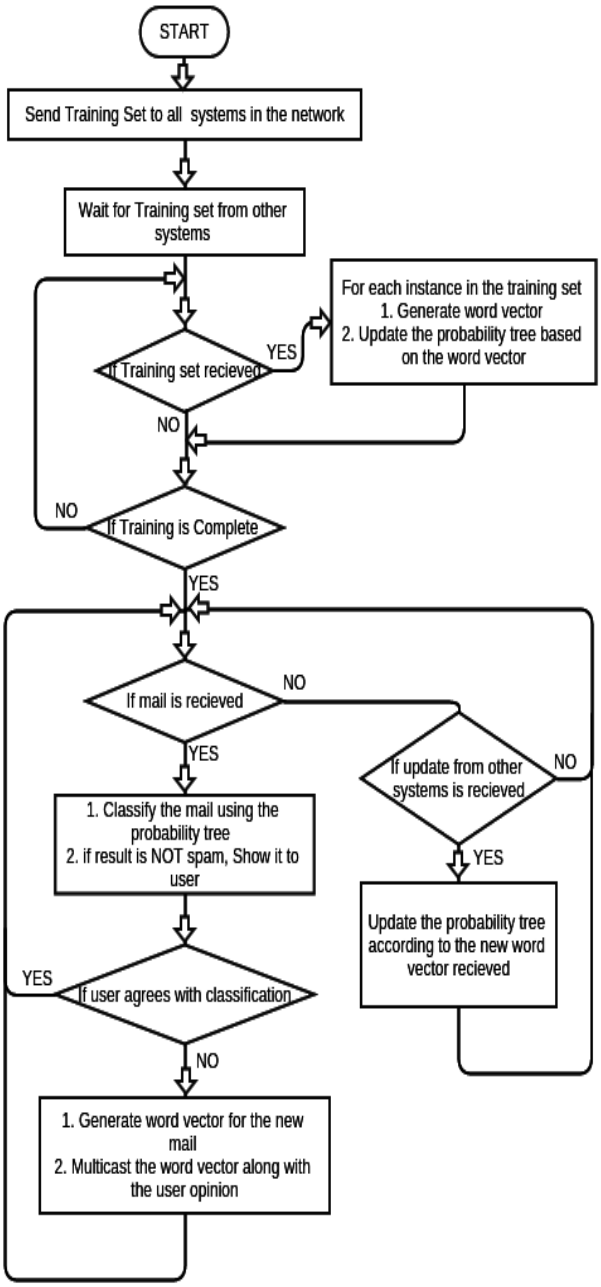

Figure 1: Flow of algorithm : Using Multicast-Update

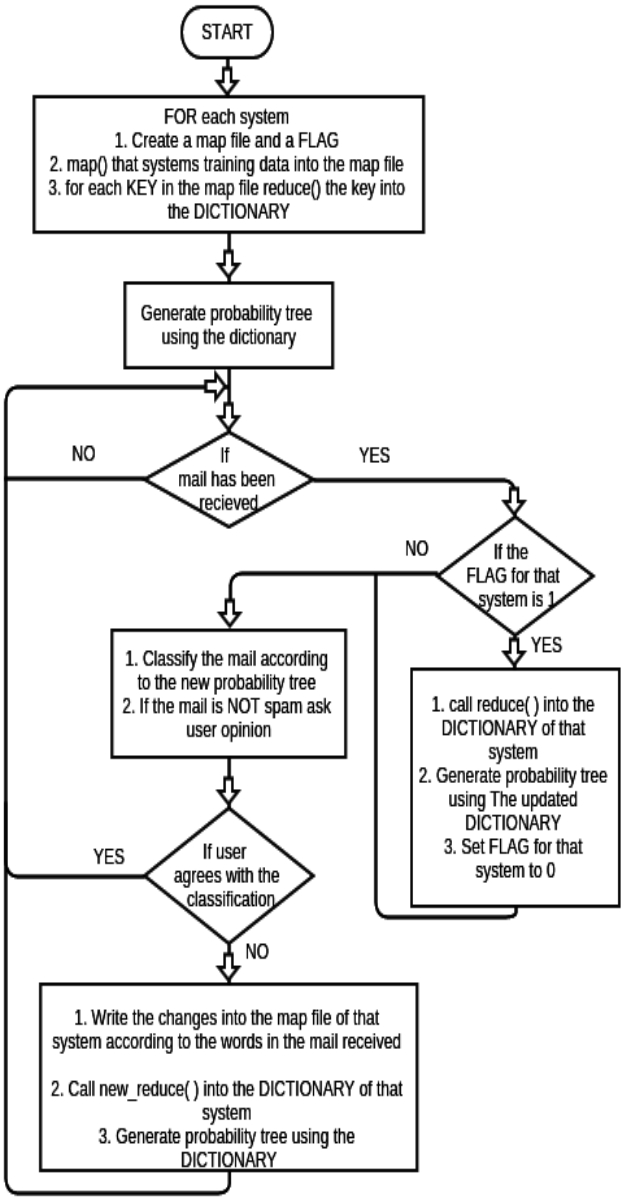

Figure 2: Flow of algorithm : Using MapReduce

different system in the group. Then, for the system where we called the $new\_reduce()$ function, we would have to regenerate the probability model and store the result in "pt" again. After all this, we would have to set the $dirty - flag$ in this system's mapfile to 0.

If the initial check of the $dirty - flag$ shows us that the flag is set to 1, we would have to run the $reduce()$ function initially, store the new key,value pairs in "r" and regenerate the probability model "pt" based on this new "r". Only after this is done should we classify the mail. This is because, as we have mentioned earlier, we require "pt" to be the newest version of itself for the cognition facility. Note here, that if the mapfile's $dirty - flag$ is 1, it implies that the actual probability model is now different and the $reduce()$ operation needs to be run again to update it.

6 ANALYSIS OF THE ALGORITHMS

6.1 Performance Analysis for the Hadoop Algorithm

The Processing Power

Increasing the processing power by two reduces the runtime to half the original if there was only one system involved. But, with a Hadoop Implementation, the relation between the speed-up and the processing power isn't that trivial. The main equation we need for understanding the Hadoop implementation [25] is the one we get from

$$runtime = \frac{overhead}{(1 - time\_to\_process\_one\_hour\_of\_data)} \quad (2)$$

From this, we get the following results: If we increase the processing power by a certain factor, say n, then our speed-up will be

$$speedup = \frac{old\_runtime}{new\_runtime} \quad (3)$$

Say the old processing power is p. This gives us

$$speedup = \frac{n - nx}{n - x} \quad (4)$$

$\lim_{n\to\infty}((n - nx)/(n - x))$
$= \lim_{n\to\infty}((1 - x)/(1 - (x/n)))$
$= 1 - x$

Graphing the above equation for two different values of n, 2 and 9, we find that as n grows, the graph moves closer and closer to the line $y = 1 - x$. This is easily proved, as the limit of the speed-up when n approaches infinity is $1 - x$. It just means that the speed up with respect to the number of systems decreases relatively as the number of systems increase.

The Aggregate Overhead

This refers to any non changing factor like network delay and other bottlenecks during the run time. If the system is optimized it will lead to a reduction in overhead. However the Overhead cannot be made zero. From the equation above, we also find that the runtime is directly proportional to the overhead. Hence, increasing the overhead by a factor of n produces a speedup of the factor $1/n$.

These are the considerations we need to make while building our system with a goal of making it time-efficient.

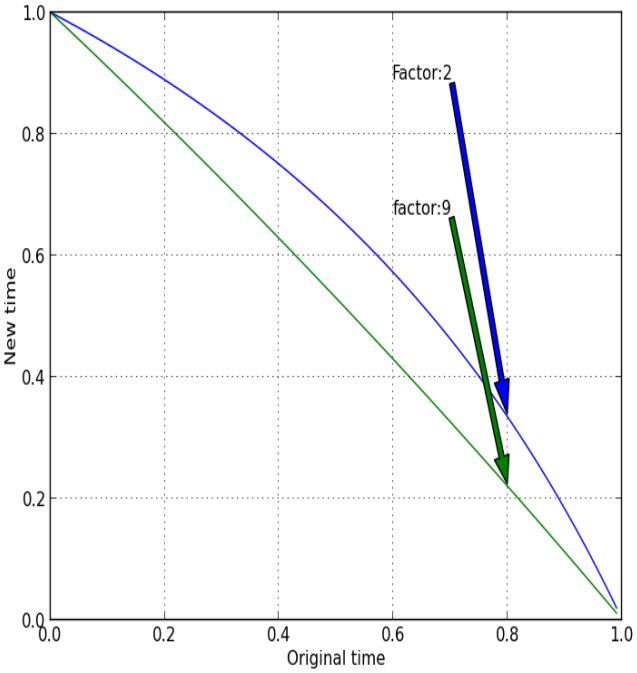

Figure 3: Variation in processing times with the processing power factor in hadoop

6.2 Performance Analysis for Multicast-Update Algorithm

In a multicast system the message to a particular group is conveyed along the minimum spanning tree for that group where the tree contains all the nodes involved in the group and the distance from the source to a node is the shortest path. We had discussed before that the network overhead is the only overhead involved in the Multicast-Update Algorithm. Thus as the networking delay increases or the amount of data increases the performance of the system with reference to time decreases. The equation below gives the delay factor for a multicast system [23].

The Overall delay(D) for a multicast message when it is sent from a sender node(s) to a group(M_G) is given as the summation of overall delay for each link of a spanning tree from the sender nodes (s) to all receivers ($r \in M_G$) and the delay for each link of the intermediate routers. Hence, this gives us to the expression for Overall delay (D) for multicast messages transmitted from sender node (s) to a receiver node (r):

$$D_{s-M_G} = \sum_{i=1}^{Z} D(L_i) + \sum_{i=1}^{n} D(L_i) \quad (5)$$

where Z is the number of receiver nodes in one group of a spanning tree (T) where n gives the total number of links a route has.

If we consider that we have a route within a spanning tree (T) from sender (s) to a receiver (r) as $R_T(s,r)$, then the multicast messages transmitted from a sender node to a receiver have a total delay of:

$$D_{<s\rightarrow(r\in M_G)>} = \sum_{L_{n,Z}\in R_T(s,r)} D(L_{n,Z}) \qquad (6)$$

where $L_{n,Z}$ gives the total links (i.e., $Z \in R_T$) that a particular message has to traverse for it to reach the specific receiver r in a route of R_T with in the Spanning Tree T as well as the links from sender s to a group M_G. From the equation when the number of systems increase the delay increases as the number of links increases and the delay is the summation of delay per link [23].

7 Simulation Results and Discussion

The simulation was carried out using the YARN simulator for hadoop. The files of the individual systems are taken as an HDFS, this file system stores the training data and the updated data(as the mails are received) of the spam filtering system. Initially it contains just the training data for each system, in their respective files. Later as the user provides feedback on the wrong classifications, the dataset is updated. Using this framework the $map()$ and $reduce()$ tasks are run, such that after the $reduce()$, the dictionary contains the word vector for the whole training set. Using this dictionary the probability model is built using the classifier algorithms. The probability model is then updated, with every wrongly classified mail.

7.1 Performance analysis for the MapReduce Framework

7.1.1 Comparison of the algorithms with flag and without flag

The $new\_reduce()$ method was introduced to remove redundant $reduce()$ and file I/O operations. This would go on to reduce the total time taken by the algorithm. The aim was to reduce extra overhead of maintaining a distributed file system for a collective learning framework. The experiment was conducted by splitting the same dataset into different number of file systems, so as to determine the impact of number of file systems on the performance. Results were taken for the number of systems ranging from 2 to 10. The number of false positives where, the user would propose a correction to the existing model was set to 30 percent of the dataset. Data is obtained for two methods one with the flag and one without the flag. The flag as discussed earlier is used to check redundant operations. The code was also run for a normal Mapreduce framework which does not use the flag and results were obtained.

Table 2 gives the values of the criterion considered for the MapReduce framework. It also gives the percentage reduction in the values for the two methods *with flag* and *without flag*. The algorithm with the $new\_reduce()$ method, i,e. the method which uses the flag to check for updated files, clearly performs better than the method which does not use the flag.

This clearly shows that introducing the flag into the HDFS does help in reducing the total time taken. The percentage reduction in the overhead increases initially and peaks at a distribution of 6 systems. It later goes to decrease as the number of systems increase. This is because even though the $new\_reduce()$ methods succeeds in reducing the overhead to a certain extent, the increase in the number of systems increases the number of $map()$ and $reduce()$ tasks considerably. This increase in the tasks due to the number of systems decreases the overall percentage reduction.

However, the Percentage reduction in File I/O changes slightly with the change in number of systems. This is because the dataset remains the same size and the only reads that are performed extra, are the reads that are performed for reading the flags of each of the system, which again increases with increase in the number of systems.

7.1.2 Impact of False Positives on the Performance

Another experiment to analyze the impact of the number of false positives, i,e. the number of times the user provides a correctional feedback to the classifier was conducted. In the simulation conducted the number of times the $new\_reduce()$ would be called(the No. of False Positives) was artificially set. The data is given for different percentage of false positives, with respect to the dataset. Table 3 gives the results.

The percentage increase given for each case is with respect to the initial 10 percent scenario. There is a clear increase in the MapReduce tasks and also the file I/O performed. This shows that providing an accurate and a rich initial training dataset would reduce the overhead later caused by the corrections suggested.

This goes to shows that the algorithm proposed is optimal for a network having systems whose individual spam and not spam preferences are similar, else the number of false positives are increased because each user has a different opinion on the spamcity of the mail. Thus showing that the personalizing the spam filter for individual systems which have radically different preferences regarding spam cannot be carried out and will result in a collective filter with poor performance with regard to both accuracy and time complexity. However, it is optimal for a network of individual systems with a similar preference. The rate of degradation is given by table 3.

Table 2: Details of the MapReduce jobs for $reduce$(without flag) and new_reduce(with flag)

Criteria	No. of bytes read	No. of read operations	Total time spent by map tasks (vcore-seconds)	Total time spent by reduce tasks (vcore-seconds)	Total megabyte-seconds taken by all map tasks	Total megabyte-seconds taken by all reduce tasks
2 systems : without flag	44590406	481	69962	9176	71641088	9396224
2 systems : with flag	44574022	479	28383	8721	29064192	8930304
percentage decrease	**0.03%**	**0.4%**	**59.4%**	**4.95%**	**59.4%**	**4.95%**
3 systems : without flag	45589933	606	101937	13728	104383488	14057472
3 systems : with flag	45508013	596	66717	9485	68318208	9712640
percentage decrease	**0.179%**	**1.65%**	**34.5%**	**30.9%**	**34.5%**	**30.9%**
4 systems : without flag	46448619	714	142729	9689	146154496	9921536
4 systems : with flag	46161899	679	108720	8050	111329280	8243200
percentage decrease	**0.617%**	**4.9%**	**23.8%**	**16.9%**	**23.8%**	**16.9%**
5 systems : without flag	47047486	790	161958	12277	165844992	12571648
5 systems : with flag	46981950	782	138825	8957	142156800	9171968
percentage decrease	**0.139%**	**1.01%**	**14.2%**	**27.04%**	**14.2%**	**27.04%**
6 systems : without flag	47744738	878	221616	18755	226934784	19205120
6 systems : with flag	47654626	867	203561	9081	208446464	9298944
percentage decrease	**0.188%**	**1.25%**	**8.14%**	**51.5%**	**8.14%**	**51.5%**
7 systems : without flag	48347949	955	230710	12638	236247040	12941312
7 systems : with flag	48241453	942	205523	10844	210455552	11104256
percentage decrease	**0.22%**	**1.36%**	**10.9%**	**14.19%**	**10.9%**	**14.19%**
8 systems : without flag	48911710	1027	241693	11389	247493632	11662336
8 systems : with flag	48801813	1013	213247	10781	218364928	11039744
percentage decrease	**0.23%**	**1.3%**	**11.7%**	**5.3%**	**11.7%**	**5.3%**
9 systems : without flag	49610583	1115	251149	20000	257176576	20480000
9 systems : with flag	49479511	1099	243915	18720	249768960	19169280
percentage decrease	**0.26%**	**1.43%**	**2.8%**	**6.4%**	**2.8%**	**6.4%**
10 systems : without flag	49903159	1154	256735	22896	262896640	23445504
10 systems : with flag	49788471	1140	250145	17300	256148480	17715200
percentage decrease	**0.23%**	**1.213%**	**2.56%**	**2.6%**	**2.56%**	**2.6%**

7.2 Performance analysis for the Classifier

7.2.1 The *unsure* class

Sometimes the mails received might be falsely classified as spam. This is a much serious problem as the user might fail to receive some important information. The mails are classified based on the spaminess score which ranges from 0 to 1, with 0 being classified as spam and 1 as not spam. When a certain incoming mail contains new words that are not there in the dictionary of the system, the mail gets a score near the vicinity of 0.5. There is a chance of that mail being classified wrongly.

To reduce these classifications, where classifier is uncertain about the nature of the mail, we can introduce an uncertain region where the spam is classified as *unsure*. The mails classified as *unsure* are sent to the user for a correct classification, this knowledge given by the user is again used to re-build the classifier model based on this corrected information. For our classifier we set the *unsure* region to be from the spaminess score of 0.45 to 0.55. This is a rather "strict" limit, and the number of false negatives or false positives would reduce and instead be classified as *unsure* as the region for the same is widened.

7.2.2 Quality of the Dataset

Inorder to ensure that initial training model built dose not develop a bias towards a certain class, we make sure that the number of instances for both *spam* and *not spam* are kept the same. This kind of problems caused by imbalanced dataset is discussed in [35]. It is also ensured that the dataset contains examples which increase the richness of the dictionary i,e. when the system is deployed, it should be made sure that the number of mails getting classified as *unsure* because of encountering a new word should be small.

The dataset should also be of a sufficiently large size. A small dataset may result in an unsure classifier i,e. when the frequency for a particular word is less, it results in a certain mail being weighted with a score which is near 0.5 or with a score which may not be accurate. Thus a quality dataset for the purpose of spam filtering, must of a sufficiently large size, balanced and should result in a dictionary with a large number of words. Additionally the pre-processing of data, like stemming [36] is

Table 3: Details of the MapReduce jobs for differing percentage of False Positives

Criteria	No. of bytes read	No. of read operations	Total time spent by map tasks (vcore-seconds)	Total time spent by reduce tasks (vcore-seconds)	Total megabyte-seconds taken by all map tasks	Total megabyte-seconds taken by all reduce tasks
10% False Positives	45686763	621	182398	8005	186775552	8197120
20% False Positives	46063595	667	189776	9610	194330624	9840640
percentage increase	**0.82%**	**7.4%**	**4.04%**	**20.04%**	**4.04%**	**20.04%**
30% False Positives	46383083	706	228892	11786	234385408	12068864
percentage increase	**1.5%**	**13.6%**	**25.4%**	**47.2%**	**25.4%**	**47.2%**

also done.

7.2.3 Accuracy of the classifier

The dataset used had three classes spam, not spam and unsure. According to the algorithms the user is shown the mail when the algorithm classified the mail as latter two classes, so as to improve the classifier model built initially. The dataset was randomly split into desired number of system files. The results on accuracy and other measures for the classifier is as follows. This result is for the scenario when the dataset was split as 4 systems and each of the individual datasets were subject to analysis.

Table 4: Comparision of Classifier performances

Criteria	System 1	System 2	System 3	System 4	Combined System
Percentage Accuracy for N-Bayes	83.5	85.7	84.3	82.8	95.4
Percentage Accuracy for SVM	51.5	52.3	61.4	57.2	88.6
No. of False Positives for N-Bayes	34	30	33	38	9
No. of False Positives for SVM	43	37	15	30	284
No. of False Negatives for N-Bayes	32	29	31	34	12
No. of False Negatives for SVM	980	968	829	881	4
No. of Unsure for N-Bayes	270	228	246	281	71
No. of Unsure for SVM.	7	5	1	3	0

As the results indicate, the naive bayesian algorithm does perform better when compared to the SVM, however this depends on the training data and the parameters over which the model is built. The parameter selection process is out of scope of this paper and the result is given to show that a bayesian classifier is the better choice of classifier for purpose of spam filtering. The results may vary if the process parameter optimization [33] is done.

We can also see that along with accuracy, the number of False Negatives and False Positives reduce greatly, when compared to the individual system output. This goes onto prove that collective learning for spam filtering is a successful idea. These results are logical, as the increase in the richness of the training dataset implies that the performance of the classifier also increases.

7.2.4 Run Time

Simulation was conducted to check the time taken for the classifier to build a model based on the training data.

Table 5: Comparision of Time Taken

Criteria	time for N-Bayesian(seconds)	time for SVM(seconds)
2000 instances	0.02	1.98
4000 instances	0.03	5.24
6000 instances	0.05	10.5
8000 instances	0.06	28.86

Table 5 gives the time taken to build a classifier model from the dataset. This data was obtained by running the classifier algorithm for the datasets of different sizes for both Naive-Bayes and SVM classifier. We can see that the time taken for Naive-Bayes Classifier almost remains the same even as the size of the dataset increases. The time taken for a particular size of dataset is also the least for Naive-Bayes classifier. Thus the use of a Naive-Bayes classifier further strengthens the concept of scalability of our spam filtering system.

The results for the SVM classifier under the fourth system also show that trend of having higher number False Negatives is reversed. This is because when you analyze the training set for the fourth system, you tend to see that there is a slightly higher number of instances which are marked spam than those marked as not spam. This is different in the training sets of the first three systems where, the number of instances marked not spam are slightly more than the number of instances marked spam. This, goes to show that, SVM develops a bias against the class whose number of instances are less. This again, is not an acceptable characteristic for a spam filter.

All these results go on to show that the Naive-bayes

classifier is a better classifier with regard to both accuracy and time complexity for the problem of spam filtering.

Thus all the results indicate that the $new\_reduce()$ method does help in reducing the overhead and also that the Naive-Bayes classifier is a better option for the purpose of spam recognition.

8 CONCLUSION

In this paper, we have addressed the issue of cognitive spam recognition using two methods, Multicast-Update and Hadoop map-reduce. The simulation results show that, the $new\_reduce()$ method reduces the time taken by MapReduce jobs by a significant amount. The reduction is maximum for a mediumly large number of systems and decreases as the number of systems increase. The results with different percentage of false positives show that the algorithm is suitable for a network where the individual systems have similar individual preferences when it comes to spam and the performance of the system deteriorates as the differences in the preference increase.

The results for the classifier performance suggest that Naive-Bayes is the better algorithm for the purpose of collective spam filtering and also verifies the concept of collective learning with an increased accuracy and reduction in the number of false positives and false negatives for the proposed combined system.

An equation giving the network delay for a network employing the Multicast-Update algorithm is given. It suggests that the network overhead increases with the increases in the number of systems. This overhead is also more than that of algorithm with the MapReduce framework. Thus, suggesting that the Multicast-Update algorithm cannot be applied for networks with large network overheads.

However, The choice of method for one's own implementation depends on how one's network is structured, the number of systems used in the network, the bottlenecks present in the network and whether they can be dealt with. Whatever the method used we do see that cognition increases the overall performance of a cluster of systems and also posses qualities like fault tolerance etc,. attributed to distributed systems. Thus proving our point that Cognition results in the betterment of spam recognition systems. The concepts proposed can be applied for any learning problem involving a network of systems with a similar aim.

The future work would involve optimizing the Hadoop algorithm, so that during the $reduce()$ operation, instead of processing the whole file it can just process the *key* values which have undergone changes since the last update. This will significantly optimize the present algorithm and thus providing an optimized framework for Learning algorithms using the current MapReduce framework.

REFERENCES

[1] Messaging Anti-Abuse Working Group. "Email metrics report", https://www.maawg.org/sites/maawg/files/ news/ MAAWG_2011_Q1-4_Metrics_Report15Rev.pdf, accessed 25th May 2015.

[2] J. Rennie, "ifile: An application of machine learning to e-mail filtering", In Proc, KDD 2000 Workshop on Text Mining, Boston, MA. August, 2000.

[3] MozillaZine, "Junk Mail Controls", https://kb.mozillazine.org/Junk_Mail_Controls, accessed 25th May 2015.

[4] M. Sahami, S. Dumais, D. Heckerman, & E. Horvitz, "A Bayesian approach to filtering junk e-mail". In Learning for Text Categorization: Papers from the 1998 workshop, Vol. 62, pp. 98-105. July 1998.

[5] S. Russell, & P. Norvig, Artificial intelligence: a modern approach, Prentice Hall, 1995.

[6] D. C. Geary, "An integrative model of human brain, cognitive, and behavioral evolution",In Acta Psychologica Sinica, Vol.39, No.3, 2007.

[7] G. F. Coulouris, J. Dollimore, & T. Kindberg, Distributed systems: concepts and design. pearson education, 2005.

[8] A. H. Wang, "DONT FOLLOW ME: Spam Detection in Twitter", In Proc. 2010 International Conference on Security and Cryptography (SECRYPT), Pg 1-10, July 2010.

[9] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, & P. Samarati, "P2P-based collaborative spam detection and filtering", In Proc. Fourth International Conference on Peer-to-Peer Computing, IEEE, pp. 176-183, August 2004.

[10] W, Dai, H. Jin, D. Zou, S. Xu, W. Zheng, L. Shi, & L. T. Yang, "TEE: a virtual DRTM based execution environment for secure cloud-end computing". Future Generation Computer Systems, Vol. 49, 2015.

[11] J. Piskorski, M. Sydow & D. Weiss, "Exploring Linguistic Features for Web Spam Detection: A Preliminary Study". In AIRWeb '08 Proceedings of

the 4th international workshop on Adversarial information retrieval on the web, ACM, pp. 25-28, 2008.

[12] Z. Yang, X. Nie, W. Xu, & Jun Guo. "An Approach to Spam Detection by Naive Bayes Ensemble Based on Decision Induction". In Proc. Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06), IEEE, pp. 861-866, October 2006.

[13] M. Sasaki & H. Shinnou, "Spam Detection Using Text Clustering". In Proc. of the 2005 International Conference on Cyberworlds (CW05), IEEE, pp. 4-pp, November 2005.

[14] V. Chang, "The business intelligence as a service in the cloud". Future Generation Computer Systems, Vol. 37, pp. 512-534, 2014.

[15] Z. Tan, U. T. Nagar, X. He, P. Nanda, R. P. Liu, S. Wang, & J. Hu, "Enhancing big data security with collaborative intrusion detection", Cloud Computing, IEEE, Vol. 1, No. 3, 2014.

[16] V. Chang & M. Ramachandran, "Towards achieving Data Security with the Cloud Computing Adoption Framework". In Press, IEEE Transactions on Services Computing.

[17] A. Garg, R. Battiti & R. G. Cascella, "May I borrow Your Filter? Exchanging Filters to Combat Spam in a Community". In Proc. 20th International Conference on Advanced Information Networking and Applications (AINA06), IEEE, pp. 5-pp, April 2006.

[18] A. G. Kakade, P. K. Kharat & A. K. Gupta, "Survey of Spam Filtering Techniques and Tools, and MapReduce with SVM". In International Journal of Computer Science and Mobile Computing, Vol.2, No. 11, 2013.

[19] R. Priyadarshini, L. Tamilselvan, "Document clustering based on keyword frequency and concept matching technique in Hadoop". In International Journal of Scientific & Engineering Research, Vol. 5, No. 5, 2014.

[20] L. Harte, Introduction to data multicasting, Althos Publishing, 2008.

[21] G. Salton, A. Wong & C. S. Yang, "A vector space model for automatic indexing", Communications of the ACM, Vol.18, No.11, 1975

[22] J. Dean, & S. Ghemawat. MapReduce: "Simplified Data Processing on Large Clusters". In Communication of ACM, Vol. 51, No.1, 2008.

[23] S. S. Rizvi, K. M. El Leithy & A. Riasat, "A Mathematical Model for Evaluating the Performance of Multicast Systems". In International Workshop on IP Multimedia Communications (IPMC 2008), IEEE, August 2008,

[24] Apache_Software_Foundation, "Apache Hadoop", http://hadoop.apache.org/, accessed 25th May 2015.

[25] Nathan Marz, "The mathematics behind Hadoop-based systems", http://nathanmarz.com/blog/the-mathematics-behind-hadoop-based-systems.html, accessed 25th May 2015.

[26] Apache_Software_Foundation. "2.3.0 (YARN)". http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html, accessed 25th May 2015.

[27] IRTF Anti-Spam Research Group, J. Levine, DNS Blacklists and Whitelists, http://tools.ietf.org/html/rfc5782, accessed 25th May 2015.

[28] J. Goodman, G. V. Cormack & D. Heckerman, "Spam and the ongoing battle for the inbox", Communications of the ACM, Vol.50, No.2, 2007.

[29] J. R. Levine. "Experiences with greylisting", In Proc. of the Second Conference on Email and Anti-Spam(CEAS), July 2005.

[30] TMDA, "Tagged Message Delivery Agent", http://www.tmda.net , accessed 25th May 2015.

[31] V. Schryver. "Distributed Checksum Clearinghouse", http://www. rhyolite.com/anti-spam/dcc, accessed 25th May 2015.

[32] V. V. Prakash & A. O'Donnell. "Fighting Spam with Reputation Systems",In ACM Queue, Vol.3, No.9, 2005.

[33] T. Bck, & H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization". Evolutionary computation, Vol.1, No.1, 1993.

[34] The Apache SpamAssassin Project, "Welcome to SpamAssassin", http://spamassassin.apache.org/, accessed 25th May 2015.

[35] J. D. Rennie, L. Shih, J. Teevan, & D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers". In Proc. International Conference on Machine Learning (ICML), pp. 616-623, August 2003.

[36] J. B. Lovins, Development of a stemming algorithm, MIT Information Processing Group, Electronic Systems Laboratory, 1968.

AUTHOR BIOGRAPHIES

Mukund. Y. R is currently an undergrad student pursuing B.tech in Computer Engineering at the National Institute of Technology Karnataka, Surathkal, Mangalore, India. His feild of interests included Machine Learning, Cryptography and Artificial Intelligence. He is also avidly interested in the field of competitive programming.

Sunil Nayak is an undergrad student pursuing B.Tech in Computer Engineering at the National Institute of Technology Karnataka, Surathkal, Mangalore, India. He has finished his fifth semester. He is interested in the fields of Artificial Intelligence, Computer Graphics and Virtual Reality. Aside from academics, he plays the Guitar and the Bass Guitar for the college's music club.

Dr. K. Chandrasekaran is currently Professor in the Department of Computer Science & Engineering, National Institute of Technology Karnataka, Surathkal, Mangalore, India, having 27 years of experience. He has more than 160 research papers published by various reputed and peer-reviewed International journals, and conferences. He serves as a member of various reputed professional societies including IEEE (Senior Member), ACM (Senior Member), CSI (Life Member), ISTE (Life Member) and Association of British Scholars (ABS). He is also a member in IEEE Computer Society's Cloud Computing STC (Special Technical Community). He is in the Editorial Team of IEEE Transactions on Cloud Computing, one of the recent and reputed journals of IEEE publication. He has coordinated many sponsored projects, and, some consultancy projects. His areas of interest - research include: Computer Communication Networks, Cyber Security and Distributed Computing and Business Computing & Information Systems Management.