SpamRank: Using Network Centrality to Filter Email Spam

Word Count: 4002

Abstract

Current spam filtering can be split into two broad categories: content and sender based. Content based filtering is relatively effective with the statistical law Bayes Theorem in Naive Bayesian filters. The Naive Bayesian filter is very effective at filtering and can also constantly update and learn how to better classify spam as more emails are passed through it. One drawback of sender based filters are their high false positive and false negative rates as well as the high maintenance required. This paper proposes a SpamRank filter, a filter that ranks senders based on their email activity, in an attempt to improve the feasibility and automation of sender based filter. The proposed SpamRank filter does successfully show that a person can be classified as a spammer or real user by looking at how many spam and ham emails a person sends and receives. However one drawback of the filter discussed in the paper, is its ability to only identify users in a given network and not account for changes in the network, in specific, new users being added.

1 Introduction

Everyone has received spam, unsolicited or undesired emails, in their email inbox at some point. Email providers all have spam filters and users can see the spam accumulating in their spam folders, but sometimes spam still makes it through. Most filters today can be classified into two categories: content based or sender based. Content based filters analyze the content, or words, in the body of the email while sender based filters look at who sent the email to determine if it is spam. Sender based filters rely on constant human input to update the list of email addresses that are known to be spam or are known to be safe. On the other hand, content based filters are constantly updating through machine learning, as the filters learn what words commonly appear in spam through their training. Although content based filters are more efficient since they do not require constant maintenance, there are some flaws in them that can be addressed with sender based filters. This paper will research the effectiveness of an automated sender based filter using eigenvector centrality.

2 Literature Review

Before researching the effectiveness of an automated sender based filter, it is important to determine its relevance. In James Veitch's TEDTalk about his replying to spam, the spam email he first receives reveals a very important problem about content based filters. The short message makes it through to his inbox because it is very short and the content could easily be in a regular email between businessmen. The only reason he knew it was spam was because of the sender and that he was not expecting any business proposals [1]. Only a sender based filter can make these human-like decisions about whether an email is spam based of a sender. As artificial intelligence improves, sender based filters could be developed that mimic how humans can simply look at the sender of an email and determine it is spam and not even have to open the email or analyze its content.

One of the most common filters today is a content based filter called a Naive Bayesian (NB) filter. During a NB filter's training, thousands of emails marked as spam or ham, which are simply emails that are not spam, are fed to the filter and the filter begins to learn what words are commonly found in spam emails. After the training period and the filter is implemented, it can constantly improve or "learn" what words are commonly found in spam based on if the user marks emails as spam. The filter follows Bayes theorem, a formula in statistics, that describes the probability of an event occurring given the probability of a prior event happening. In this case, the prior event is the probability of a word appearing in an email and the event is the probability the email is spam. There are many different ways that an NB can assign and use these probabilities, or tokens [2], [3]. Some NB filters take into account all of the words in an email while others strategically choose which words to include in their calculations based on how far a word's probability of spam deviates from neutral, thereby ignoring commonly used words like articles and pronouns that occur often in both ham and spam emails.

Two common types of NB filters are multinomial and multivariate [4], [5]. The multinomial NB filter, rather than just keeping track of the percentage of a word appearing, also takes into account the percentage of a word appearing multiple times. A traditional NB filter only looks at the percentage of a word appearing and does not keep track of how many times it appears. One flaw of this which the multinomial NB filter attempts to fix is the fact that the chances of certain key spam words appearing multiple times in a short email is very rare. A multinomial NB filter can better filter very short emails, like the one in Veitch's TEDTalk that are spam and have certain key words characteristic of spam emails. Multinomial NB, or Bernoulli NB, is similar in that it keeps track of each word's number of appearances. However, the multinomial NB treats each word as a boolean, which means having only two possible states. Rather than keeping track of a probability of spam, it marks each word as spam or ham and then looks at how many words in the whole email are spam to determine if the email is spam.

While these two types of NB filters attempt to improve the already relatively successful traditional NB filter, there are still many ways any NB filter can be attacked. As mentioned early in the TEDTalk, a flaw of NB filters is that they cannot handle emails with common words. Two common attacks are dictionary attacks and focused attacks [6], [7]. In a dictionary attack, spammers send large emails with all the words in the English dictionary. Because the receiver knows the email is spam, due to the unrecognized sender and the content, the receiver marks it as spam. However, since the spammer included every English word, common words now have a higher probability as being marked as spam, so future legitimate emails may be marked as spam. Similarly, in focused attacks, spammers know what words may appear in an email they wish to block, so they send lots of emails with those words, knowing that they will be marked as spam since they are from an unknown sender. The filter then "learns" that these words are more likely to be spam. As a result, the email the spammer wished to be blocked will have a higher chance of ending up in the spam folder where the receiver may not read it. For example, if a spammer knows of a competing business offer from a certain company for a certain product, they can send emails with the company or product name which will be marked as spam by the receiver since the sender is unrecognized. This will increase the filter's probability of the company or product names as being spam. Then, when the real email is sent including the company or product name, the email will more likely be marked as spam.

While NB and other content filters can be attacked easily, one important benefit is the minimal maintenance required. NB filters are an example of machine learning, a basic form of artificial intelligence. The filters can continue to "learn" and improve and as a result, remain relatively effective after being trained. However, sender based filters require constant maintenance. The three main types of sender based filters are black lists, white lists, and greylisting. Black lists block all emails from senders on a certain list [8], [9]. White lists are the exact opposite; they are a list of senders that only email from them are allowed through the filter. Both black lists and white lists require constant human input to maintain and update these lists. They also have low success rates with many false positives and false negatives. Black lists can easily allow spam from users that are not on the list of spammers, and white lists can easily block out anyone's email mistakenly. Greylisting is the process where filters send back an error message to all email senders. Greylisting relies on the fact the real users are more likely to try and resend the message rather than spammers, so they let emails through that have been sent twice [9]. Greylists are not that effective however and can be easily overcome since some real users may not try to resend the email and spammers who know of this tactic can send all their emails twice. Because content based filters are already relatively successful, this paper will try to see if sender based filters can be as successful. Because current sender based filters using lists have many flaws, this paper will attempt to research a method using eigenvector centrality, by looking at the relationship between senders.

Eigenvector centrality is a measure of the influence of users in a network. One of the most well known examples of eigenvector centrality is Google's PageRank. PageRank was what set Google apart from other search engines at the time of its creation. PageRank is a ranking for websites based off of their relative importance. Almost all websites have links that go to another page or website. PageRank considers these links as a "vote" for the page it links to, saying that it is a good website. Every website earns votes from other websites and then it divides its votes up equally to distribute to the websites it votes for. This system makes it so that a popular website that many websites link to is considered influential such that the websites it votes for gain more votes since a more trusted website is voting for it [10]. This system of votes and popularity is called eigenvector centrality in mathematics. It seems impossible to calculate the popularity of one website since it is based off of every other website's popularity, creating an infinite cycle. However, by recording the amount of links to and from a website in a matrix, mathematicians can calculate what is called an eigenvector, a vector that only changes by a scalar when multiplied with the matrix, meaning it stores the characteristics of the original matrix. This eigenvector gives the popularity of all websites in the case of PageRank. Centrality has been applied more broadly to spam websites [11], where centrality formulas based off of and similar to PageRank are used to identify spam websites. Other studies have used PageRank based formulas to identify the relationship and influence of users in social networks like Twitter [12]. In addition, centrality has been used to analyze the relationships between users on an email server [13]. Hardin and his group studied the relative importance of employees of the company Enron by looking at the emails they sent and received. Rather than focusing on identifying spam, Hardin and his group chose to identify which users were closer with each other and which users were more influential to see if the hierarchy of the company could be seen in their emails.

A research group led by Chirita has proposed a spam filtering method based off of PageRank by using an email received as the equivalent of what a vote would be in PageRank [14]. This idea is based off the fact that a user is most likely only to send emails to real users, so emails sent to a user can be counted as a vote for that user receiving the email, saying they are a real user. By this method, the "SpamRank" (SR) of a user is calculated based off of the emails it receives. However, this paper will try and analyze the effectiveness of a more detailed method looking at the number of emails sent with ham emails giving votes and spam emails taking away votes. Since spammers are not considered credible users in the network, the idea behind this paper's experimental SpamRank is that spammers should not essentially be given equal voting rights. Spammers' votes should not be as credible to prevent their influence in boosting the SpamRank of other spammers.

A filter using centrality is just a small step in creating an automated sender based filter. Such a filter would be a great improvement in sender based filters in that they would be more accurate and require less maintenance and resources to upkeep. A successful one could be very useful in the field of artificial intelligence in the future when it improves. When a person looks at their inbox, they can immediately tell if an email is spam without opening it. All they see is the sender and subject as well as the first few words. Filters today have to "open" the email and analyze its contents, as discussed earlier about content based filters. However, in today's growing advances in artificial intelligence, we may be able to one day create filters that classify spam like we do, simply by looking at the sender or subject. A centrality based filter is a good step towards that direction. Future research should be carried out using better artificial intelligence to try and create a fully automated and successful sender based filter. Not only will this improve spam filtering but also make advances in the field of artificial intelligence.

3 Method

An experimental method was chosen in order to test the effectiveness of the proposed filter. First off, it was not feasible for this paper to implement a developed filter on a real network due to time and resource constraints, as well as privacy issues regarding people's emails. Two other options of testing the proposed filter were using a simulation or an existing spam dataset. A simulation has the advantage of being easy to perform and calculate since you would input data into the calculation to reflect common spam attacks. However, I chose to test the proposed filter on a spam dataset in order to have a more random sample and remove any bias I might have caused by creating the data in the simulation myself, as well as testing the proposed filter against a more real-world situation. I originally considered using the Enron spam dataset because all the emails were taken from one network of users and would be a realistic small scale test for the proposed filter. However, I chose to use the CSDMC2010 spam dataset, originally used for a data mining competition from the International Conference on Neural Information Processing, because it is virus free and more easily accessible [15]. Using Microsoft Access 2016, I created a database of all the emails recording the email addresses of the senders and receivers as well as whether or not the email was spam. From there, I could compile a table of all the users in the network and record how many ham and spam emails they sent and to who as well as how many emails they received and from who. This allowed me to easily transfer my data into my filter's equation.

The proposed filter gives the ranking of a user based on the equation:

$$SR(p) = (1-d) + d \sum_{q \in O(p)} (SR(q_h) - SR(q_s))$$

$SR(p)$ represents the SpamRank of user p. d is a constant called the dampening factor which allows the SpamRank to converge on a value after performing the power iteration algorithm, which will be explained later. q represents another user in the network. $q \in O(p)$ specifies that all q must be in the set $O(p)$ which is the outbox of user p. This means that user q has received an email from user p. $SR(q)$ is the SpamRank of user q. q_h represents the users who received ham emails from user p, and similarly q_s represents the users who received spam emails from user p.

At first glance, this equation seems impossible to solve since every user's SR is dependent on every other user's SR, leading to an infinite cycle. However, it is possible with a power iteration algorithm. A power iteration algorithm is the process where all the SR of the users are originally set at the same value. For my calculations, they all started at 0.75. Then my calculator solves the equation of each user. After solving each SR, the calculator updates the list of users' SR and continues solving the equation for each user with the updated, more accurate values for SR. The calculator solves the equation for each user 1000 times. The SR will eventually converge on the actual values. The special property of the proposed equation is that only one set of SR values will make all users' SR equations true. Therefore, when you plug in numbers, the equation will always decrease or increase the SR until it converges on its actual value.

I then transferred all my data I recorded from the emails into a calculator I programmed in Java, shown in Appendix B. After calculating the SR of each email user, this ranking would be compared to a threshold value, which I determined to be $1-d$. I chose this value because it is the initial SR of all users. If the user's SR value falls below the threshold, the sender is most likely a spammer and the email would be marked as spam. I could compare this to the results of current NB filters as well as Chirita's/PageRank filter.

4 Results

I only analyzed the first 73 emails in the CS-DMC2010 corpus due to time limitations. This network of users is considered a sparse network, meaning there is very little relationship between users. In this situation, the relatinoship between usrs in this network is represented by an email sent or received. There are small groups of users that are related, often just two users. Because there were 73 emails, there potentially could be 146 distinct users. This would be the sparsest this network could possibly be. However in this set, there were 112 distinct users. Thirteen users sent or received more than one email. After inputting all my data into my calculator, I recorded all of the SRs calculated for the users. I also recalculated all the SR values using Chirita's PageRank formula. I compared the results of both filters. For the proposed filter, I used a threshold of $SR < 0.15$ to indicate spam, while for Chirita's/PageRank formula, I used a threshold of $SR \leq 0.15$ to indicate spam. Please see appendix A for full results of the calculated SpamRanks. The results of the filters' success rates can be seen in the table below.

Filter Method	Chirita/ PageRank	Experimental SpamRank	NB Filter
Users Successfully Identified	63/112	112/112	■
Success Rates	56.25%	100%	90-99% [2], [3]

Table 1: Success Rates of Filters

Given the results of this research, we can see that a traditional PageRank is lacking an important aspect since its success rate is only 56.25%. On the other hand, the proposed SpamRank filter was able to identify 100% of the users correctly, which is comparable to today's content based filters such as the NB filter. Most of the content based filters analyzed by the sources in my literature review had success ratings in the upper 90%.

One flaw of a simple PageRank formula filter is that it cannot distinguish between a spammer who sent one email to a user and another user who sent one email to another user. To the filter, both situations are identical since the filter only analyzes emails received. The filter only sees that the two users received one email from a user who has not received any emails. This is why many of the users are incorrectly classified for the traditional PageRank since they all are at the default SR value of 0.15. This is also why I set the threshold values for both differently.

On the other hand, the experimental SpamRank filter had a 100% success rate in identifying all 112 users correctly. Because it calculated SpamRanks in more detail by accounting for the spam and ham emails a user sent and received, it did not have the same flaw as the traditional PageRank filter since spammers' SpamRank were decremented as their emails were marked as spam.

While the proposed filter had a 100% success rate at classifying a known network of users, it does not account for users new to the SR system. The filter only looks at one instance of all existing users in the network and identifies them as spammers or real users. The first time a user sends an email, the filter has no data to calculate the user's SR. If a new user sends an email, the proposed filter would let it pass because the user has a starting SR of 0.15 which does not fall into the proposed threshold of $SR < 0.15$. Accounting for this, the proposed filter would have misclassified 28 emails, giving it a success rate of 75%. It would have let all 28 spam emails pass. I chose this threshold because users who receive ham email from other actual users will have a SR of 0.15 until they send an email. However unlike Chirita's PageRank filter, the proposed filter ensures this does not happen for spammers by decreasing their SR as more users mark their emails as spam. Rather than changing the threshold, a method of filtering new users needs to be investigated.

5 Conclusion

Based on the results, the proposed filter does successfully identify which existing users in a network are spammers. A centrality based filter seems possible to achieve a more automated sender based filter. One flaw noted about Chirita's traditional PageRank filter is it fails to distinguish between spammers and users that have sent the same amount of emails and received no emails. While the proposed experimental SpamRank does account for this, additional research needs to be performed on how the filter will deal with new users added to the system.

There are many possible ways to account for new users added. One very promising possibility is to combine both the proposed filter and Chirita's PageRank filter and add both SR to obtain a final SR. Since the proposed filter accounts for users that have not sent any emails and Chirita accounts for users that have not received any emails, both filters combined could account for both. A new threshold could be set at $SR < 0.3$ indicating spam. Another method is to require users to receive an email from another user before being able to send an email. The new user's SR could then be calculated using the SR of the user they received an email from. Lastly, a traditional NB filter could be used to filter the first email sent of every user. These methods seem promising in creating a completely successful centrality based filter but require additional research.

There are also many different variables regarding the spam dataset used that may have affected the success rates. Additional research should be performed with more emails used which will result in a larger network of users. Due to time constraints, this paper could not do so. However, with access to more advanced computer programs that could analyze emails, extract the necessary information, and record the data, more analysis could have been performed. Due to resource constraints and potential privacy issues, research could not be performed on real networks, which would give more accurate, real world data. Future research should be performed on real networks. One noticeable trend in the results is that the spammers only sent one email to another user. This is because this dataset does not look at all the emails a user sends but rather just a sample of emails sent within a certain time frame. Therefore, it does not account for bulk spam, which is very common for spammers to send a spam email out to many users. Research performed on a real network could account for many more accurate spammer tactics and can give a more accurate success rate about whether or not centrality filters are feasible.

References

[1] J. Veitch, "This is what happens when you reply to spam email," in *TEDGlobal>Geneva*, 2015.

[2] V. P. Deshpande, R. F. Erbacher, and C. Harris, "An evaluation of naive bayesian anti-spam filtering techniques," *IEEE*, 2007.

[3] W. S. Yerazunis, "The spam - filtering accuracy plateau at 99.9% accuracy and how to get past it,"

[4] J. Eberhardt, "Bayesian spam detection," 2015.

[5] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with naive bayes - which naive bayes?," in *CEAS*, 2006.

[6] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. A. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," in *LEET*, 2008.

[7] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *CEAS*, 2005.

[8] J. Jung and E. Sit, "An empirical study of spam traffic and the use of dns black lists," in *Internet Measurement Conference*, 2004.

[9] G. V. Cormack, "Email spam filtering: A systematic review," *Foundations and Trends in Information Retrieval*, vol. 1, pp. 335–455, 2006.

[10] "The pagerank citation ranking: Bringing order to the web," 1998.

[11] Z. Gyöngyi, H. Garcia-Molina, and J. O. Pedersen, "Combating web spam with trustrank," in *VLDB*, 2004.

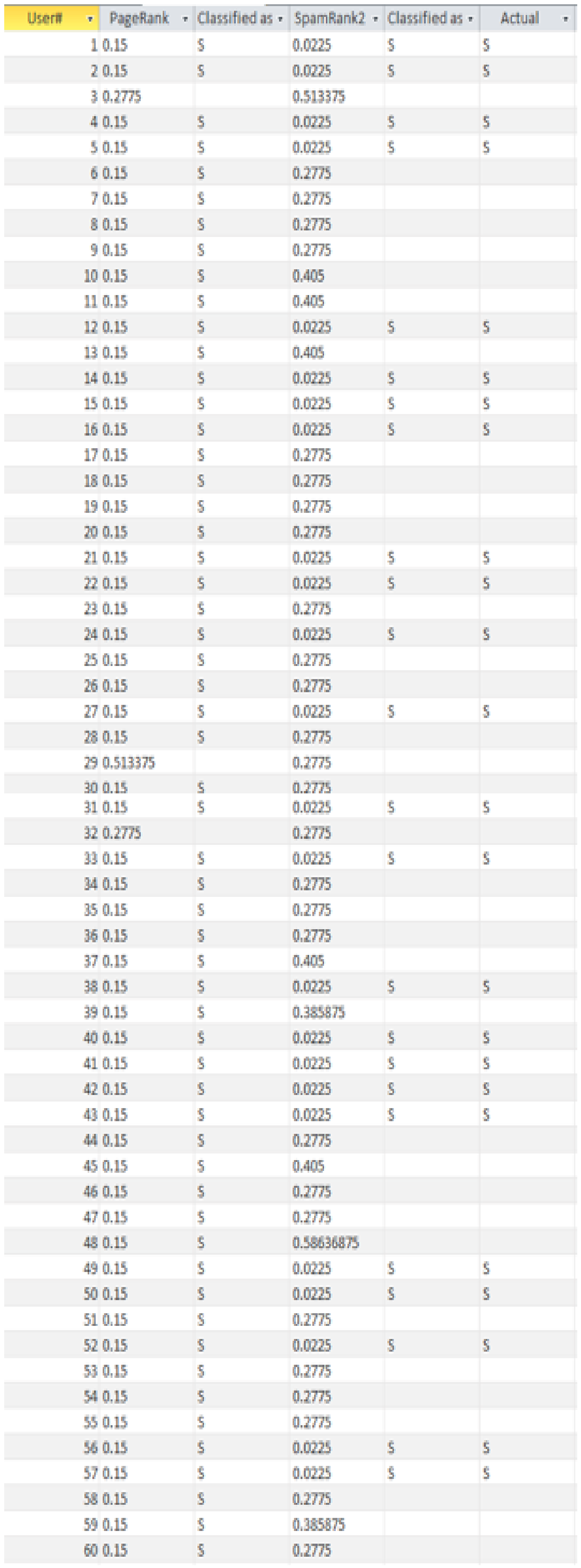
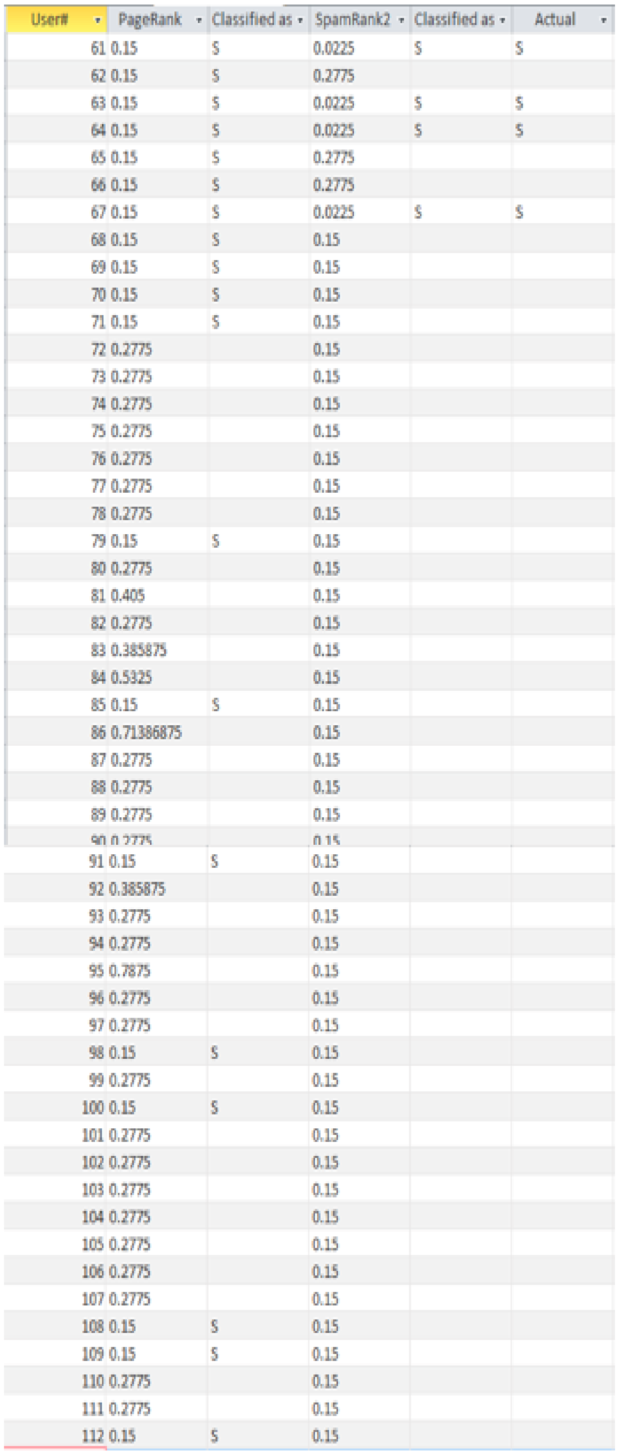
[12] J. Song, S. Lee, and J. Kim, "Spam filtering in twitter using sender-receiver relationship," in *RAID*, 2011.

[13] J. Hardin, G. Sarkis, and P. C. Urc, "Network analysis with the enron email corpus," *CoRR*, vol. abs/1410.2759, 2014.

[14] P.-A. Chirita, J. Diederich, and W. Nejdl, "Mailrank: using ranking for spam detection," in *CIKM*, 2005.

[15] "Csmining group." `http://csmining.org/index.php/spam-email-datasets-.html`.

Appendix A This is the table of calculated SpamRank values.

User#	PageRank	Classified as	SpamRank2	Classified as	Actual
1	0.15	S	0.0225	S	S
2	0.15	S	0.0225	S	S
3	0.2775		0.513375		
4	0.15	S	0.0225	S	S
5	0.15	S	0.0225	S	S
6	0.15	S	0.2775		
7	0.15	S	0.2775		
8	0.15	S	0.2775		
9	0.15	S	0.2775		
10	0.15	S	0.405		
11	0.15	S	0.405		
12	0.15	S	0.0225	S	S
13	0.15	S	0.405		
14	0.15	S	0.0225	S	S
15	0.15	S	0.0225	S	S
16	0.15	S	0.0225	S	S
17	0.15	S	0.2775		
18	0.15	S	0.2775		
19	0.15	S	0.2775		
20	0.15	S	0.2775		
21	0.15	S	0.0225	S	S
22	0.15	S	0.0225	S	S
23	0.15	S	0.2775		
24	0.15	S	0.0225	S	S
25	0.15	S	0.2775		
26	0.15	S	0.2775		
27	0.15	S	0.0225	S	S
28	0.15	S	0.2775		
29	0.513375		0.2775		
30	0.15	S	0.2775		
31	0.15	S	0.0225	S	S
32	0.2775		0.2775		
33	0.15	S	0.0225	S	S
34	0.15	S	0.2775		
35	0.15	S	0.2775		
36	0.15	S	0.2775		
37	0.15	S	0.405		
38	0.15	S	0.0225	S	S
39	0.15	S	0.385875		
40	0.15	S	0.0225	S	S
41	0.15	S	0.0225	S	S
42	0.15	S	0.0225	S	S
43	0.15	S	0.0225	S	S
44	0.15	S	0.2775		
45	0.15	S	0.405		
46	0.15	S	0.2775		
47	0.15	S	0.2775		
48	0.15	S	0.58636875		
49	0.15	S	0.0225	S	S
50	0.15	S	0.0225	S	S
51	0.15	S	0.2775		
52	0.15	S	0.0225	S	S
53	0.15	S	0.2775		
54	0.15	S	0.2775		
55	0.15	S	0.2775		
56	0.15	S	0.0225	S	S
57	0.15	S	0.0225	S	S
58	0.15	S	0.2775		
59	0.15	S	0.385875		
60	0.15	S	0.2775		
61	0.15	S	0.0225	S	S
62	0.15	S	0.2775		
63	0.15	S	0.0225	S	S
64	0.15	S	0.0225	S	S
65	0.15	S	0.2775		
66	0.15	S	0.2775		
67	0.15	S	0.0225	S	S
68	0.15	S	0.15		
69	0.15	S	0.15		
70	0.15	S	0.15		
71	0.15	S	0.15		
72	0.2775		0.15		
73	0.2775		0.15		
74	0.2775		0.15		
75	0.2775		0.15		
76	0.2775		0.15		
77	0.2775		0.15		
78	0.2775		0.15		
79	0.15	S	0.15		
80	0.2775		0.15		
81	0.405		0.15		
82	0.2775		0.15		
83	0.385875		0.15		
84	0.5325		0.15		
85	0.15	S	0.15		
86	0.71386875		0.15		
87	0.2775		0.15		
88	0.2775		0.15		
89	0.2775		0.15		
90	0.2775		0.15		
91	0.15	S	0.15		
92	0.385875		0.15		
93	0.2775		0.15		
94	0.2775		0.15		
95	0.7875		0.15		
96	0.2775		0.15		
97	0.2775		0.15		
98	0.15	S	0.15		
99	0.2775		0.15		
100	0.15	S	0.15		
101	0.2775		0.15		
102	0.2775		0.15		
103	0.2775		0.15		
104	0.2775		0.15		
105	0.2775		0.15		
106	0.2775		0.15		
107	0.2775		0.15		
108	0.15	S	0.15		
109	0.15	S	0.15		
110	0.2775		0.15		
111	0.2775		0.15		
112	0.15	S	0.15		

Appendix B

The following is the code I wrote in Java to perform the power iteration algorithm to generate the SpamRank of the users:

```
import java.util.Scanner;

public class SRCalculator
{
   public static void main(String[] args)
   {
      Scanner amt = new Scanner(System.in);
      System.out.print("Enter number of users: ");
      int amount = amt.nextInt();

      double[] userSpamRank = new double[amount];
      for (int k=0; k<userSpamRank.length; k++)
         {userSpamRank[k]=0.75;}

      user[] users;
      users = createLinks(amount);

      for(int k=0; k<1000; k++)
      {
         for(int a=0; a<amount; a++)
         {
            userSpamRank[a] = solveSR(a, amount, userSpamRank, users);
            System.out.print(userSpamRank[a]+" ");
         }
         System.out.println(" ");
      }
      System.out.println(" ");

      for(int b=0; b<amount; b++)
         { System.out.println("User "+(b+1)+": "+userSpamRank[b]); }
   }


   public static double solveSR(int uN, int a, double[] uR, user[] x)
   {
      int[] tempHOutbox = x[uN].getHamOutbox();
      int[] tempSOutbox = x[uN].getSpamOutbox();
      double hSR=0.0;
      double sSR=0.0;

      if(tempHOutbox.length!=0)
      {
         for(int k=0; k<tempHOutbox.length; k++)
         {
            hSR+=uR[tempHOutbox[k]];
         }
      }

      if(tempSOutbox.length!=0)
      {
         for(int k=0; k<tempSOutbox.length; k++)
         {
            sSR+=uR[tempSOutbox[k]];
         }
      }
      double temp;
      temp = (0.15) + ((0.85*(hSR-sSR)));

      return temp;
   }

   public static user[] createLinks(int x)
   {
      user[] people = new user[x];
      for (int k=0; k<x; k++)
      {
         System.out.println(" ");
         System.out.println("For user "+ (k+1));
         user temp = new user();
         people[k]=temp;
      }
      return people;
   }
}

public class user
{
   private int[] inbox;
   private int[] hamOutbox;
   private int[] spamOutbox;

   public user()
   {
      hamOutbox=setHamOutbox();
      spamOutbox=setSpamOutbox();
      inbox=setInbox();
   }

   public int[] getHamOutbox() {return hamOutbox;}
   public int[] getSpamOutbox() {return spamOutbox;}
   public int[] getInbox() {return inbox;}

   public int[] setHamOutbox()
   {
      Scanner ns = new Scanner(System.in);
      System.out.print("Enter number of ham emails sent: ");
      int numSent = ns.nextInt();

      int[] temp = new int[numSent];

      for (int k=0; k<numSent; k++)
      {
         Scanner ws = new Scanner(System.in);
```

```
            System.out.print("Enter user number ham email was sent to: ");
            int whoSent = ws.nextInt()-1;
            temp[k]=whoSent;
        }

        return temp;
    }

    public int[] setSpamOutbox()
    {
        Scanner ns = new Scanner(System.in);
        System.out.print("Enter number of spam emails sent: ");
        int numSent = ns.nextInt();

        int[] temp = new int[numSent];

        for (int k=0; k<numSent; k++)
        {
            Scanner ws = new Scanner(System.in);
            System.out.print("Enter user number spam email was sent to: ");
            int whoSent = ws.nextInt()-1;
            temp[k]=whoSent;
        }

        return temp;
    }

    public int[] setInbox()
    {
        Scanner nr = new Scanner(System.in);
        System.out.print("Enter number of emails received: ");
        int numReceived = nr.nextInt();

        int[] temp = new int[numReceived];

        for (int k=0; k<numReceived; k++)
        {
            Scanner ws = new Scanner(System.in);
            System.out.print("Enter user number email was sent from: ");
            int whoSent = ws.nextInt()-1;
            temp[k]=whoSent;
        }

        return temp;
    }
}
```