A Multiagent-based Peer-to-Peer Network in Java for Distributed Spam Filtering

Jörg Metzger, Michael Schillo, and Klaus Fischer

German Research Center for Artificial Intelligence,
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{jmetzger, schillo, kuf}@dfki.de
http://www.dfki.de/

Abstract. With the growing amount of internet users, a negative form of sending email spreads that affects more and more users of email accounts: Spamming. Spamming means that the electronic mailbox is congested with unwanted advertising or personal email. Sorting out this email costs the user time and money. This paper introduces a distributed spam filter, which combines an off-the-shelf text classification with multiagent systems. Both the text classification as well as the multiagent platform are implemented in Java. The content of the emails is analyzed by the classification algorithm 'support vector machines'. Information about spam is exchanged between the agents through the network. Identification numbers for emails which where identified as spam are generated and forwarded to all other agents connected to the network. These numbers allow agents to identify incoming spam email. In this way, the quality of the filter increases continuously.

0.1 Keywords

Multiagent Systems, Peer-To-Peer Networks, Spam Filter, Text Categorization with Support Vector Machines.

1 Introduction

One of today's greatest problems of email traffic is the lack of authentification of email servers and senders. Without the possibility to identify the sender of an email, there are people who take advantage of this security gap. They send spam email to a huge amount of people, who did not solicited them. The cost of sending spam emails is very low, giving an incentive to force a high number of users to spend time (and online fees) to filter unwanted messages. There are several ways to get email addresses. Spammers, the sender of spam email, scan mailing lists, homepages and forums. This is done with the help of spambots, programs which automatically scan the internet for email addresses. These lists of valid email addresses are sold to other spammers which use them for their own purposes. Statistics about the amount of spam received since 1996 have shown that in the last two years, the amount of spam email has increased exponentially [10]. These

statistics are representative for most users of email accounts. There is no global legal protection from this problem. On the server side, the configuration of your mailserver can be configured to sort out email according to certain criteria and to block email from certain domains. For effective protection, regular reconfiguration of the server settings in necessary. There is also the risk of deleting email which is not spam. So this is not a satisfactory solution. For this reason, a lot of software has been developed to recognize and filter spam email. An frequently used approach is the creation of rules to filter emails. Programs like RIPPER possess a database of rules [1]. It is difficult to adapt these rules to changes of the email content and to keep them up-to-date. Other spam filters [6, 13] work with algorithms developed for text classification. Although these filers delete a huge amount of spam, they only work isolated from other filters. So they cannot give information about spam to other filters to upgrade their knowledge on new forms of spam. There exists only one commercial distributed spam filter [12] which exchanges spam information over a network in a distributed fashion. The Vipul's Razor filter agents connect to servers which maintain a database with identification numbers for spam emails. They identify spam emails with the help of the identification numbers. Although this filter system is basically comparable with our spam filter network, it has several disadvantages. Our network has no need of centralized servers. Instead, antispam agents are connected as nodes of a peer-to-peer network which can communicate directly with each other to exchange information. Furthermore, the Vipul's Razor filter agents cannot deal with a new form of spam which more and more often appears.
With this work, we introduce a network of spam filter agents which combines the power of text classification algorithms with a peer-to-peer network. The following section describes the basics of text classification and multiagent systems on which we base our work. In Section 3 we introduce our spam filter network and show how it has been implemented in Java. The paper closes with summarizing remarks and future lines of research.

2 Basics

2.1 Support Vector Machines

Users of our filter network should have the opportunity to let the spam filter classify incoming email into the categories *spam* or *nonspam* automatically. The user can correct the classification result if needed. Today, there exits a huge number of classification algorithms to choose from. Several comparisons of the different methods have demonstrated that the classification algorithm Support Vector Machines (SVM) outperforms most of the other algorithms (for example Naive Bayes and k-nearest neighbor) [2, 5] and reaches rates of accuracy in classifying spam over 95 percent [11]. In order to classify text, its content has to be represented as a feature vector. We choose a word as feature. The corresponding vector is composed of various words from a dictionary formed by analyzing the text contents. The weighting of the feature vector is constructed as follows. If a particular word occurs in the content of the text, the corresponding value of

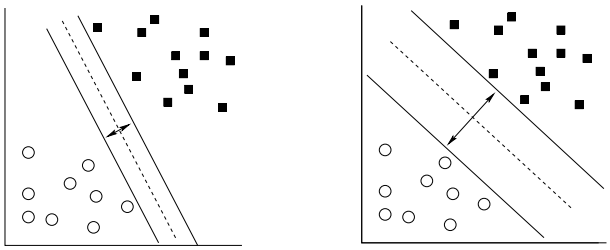

Fig. 1. SVM construct the hyperplane which maximizes the border between the two classes. The classifier on the right is the better one.

the feature is 1, otherwise 0. This is called the 'binary representation' of the feature vector. These vectors can be described as data points in an n-dimensional space. The basic idea of SVM is to find a hyperplane which best separates the data points into two classes. In this way, the number of classification errors is minimal. Figure 1 shows the best separating hyperplane between the circles and squares in the two-dimensional space.

More precisely, the decision surface by SVM for linearly separable data points is a hyperplane which can be written as $w^T x - b = 0$. x is an arbitrary data point to be classified and the vector w (vector of the hyperplane) and the scalar b are learned from the training data. Let $D = \{(x_i, y_i)\}$ denote the training set with feature vector $x_i \in R^N$ and $y_i \in \{-1, 1\}$ be the classification for x_i ($y_i = +1$ if $x_i \in Spam$ and $y_i = -1$ if $x_i \in Nonspam$), the SVM problem is to find w and b that satisfy the following constraints

$$\begin{aligned} w^T x_i - b \geq 1 \quad if\ y_i = 1, \\ w^T x_i - b \leq -1 \quad if\ y_i = -1 \end{aligned} \tag{1}$$

and that the vector 2-norm of w is minimized (maximizing the separating hyperplane). Training examples that satisfy (1) are termed support vectors. The support vectors define two hyperplanes, which both go through the support vectors of the respective class. This quadratic optimization problem can be efficiently solved and a new vector x^* can be classified as follows

$$f(x^*) = sign\{w^T x^* - b\} \quad with \quad w = \sum_{i=1}^{N} v_i x_i \tag{2}$$

The course of the hyperplane is only determined by the positions of the N support vectors x_i with weighting v_i which the algorithm calculates.

2.2 Multiagent Systems and FIPA-OS

Multiagent Systems (MAS) consist of several autonomous agents, which work independently and distributed [14]. For that purpose several interfaces are defined.

The MAS makes available basic services, where communication and interaction protocols are specified. The agent platform we use is fully implemented in Java. The FIPA Open Source (FIPA-OS) platform [7] is continuously improved as an open source project and conforms to the agent standards set up by the Foundation for Intelligent Physical Agents (FIPA) [3]. FIPA-OS provides 'white pages' and 'yellow pages' services for the agents. The Directory Facilitator (DF) is a special agent which mediates between all agents on the platform. This includes tasks like registration, service information on other agents, deregistration etc. An index of the names of agents which are currently registered with the agent platform is maintained by the agent management service (AMS) and can be accessed by the DF. The Message Transport System (MTS) is responsible for the platform to platform transport and encoding of the messages. The agents of the MAS work together to achieve the common goal of spam filtering. In order to exchange information about spam, the agents must interact together at a semantically rich level of discourse. FIPA-OS provides an agent communication language (ACL) which describes a standard way to package messages, in such a way that it is clear to other compliant agents what the purpose of the communication is. These rules ensure that the semantic integrity of the language is retained. Negotiation, cooperation and information exchange are supported.
In the next section, we will show how the combination of multiagent systems and text classification leads to a functional distributed spam filter network.

3 Structure of the Distributed Spam Filter

In the following, we describe our novel approach on spam filtering based on the advantages of 'support vector machines' and multiagent systems. We explain the design and the implementation of a peer-to-peer spam filter network. Within the network, data concerning spam is exchanged between the agents. For each email, an identification number is created as described in Section 3.1. If the email is identified as spam by the classification algorithm, the corresponding number is sent to all other agents connected to the network. The antispam agents compare the identification numbers of all incoming email with those spam numbers collected from other agents. The concrete procedure of the antispam agents is described in Section 3.2. The last Section shows the interface in detail and the possibilities available for the user to influence the classification of email and to correct false decisions made by the spam filter.

3.1 Creation and Comparing of Hash Values

One of the necessary functions of the antispam agent is the ability to compare two different emails. For reasons of security, it is not desirable to send the plain text of the content of emails over to network to other agents and compare them word by word. The user certainly does not want other users to be able to reconstruct the contents of the received email from the data sent. Representations of the email content like word vectors cannot be sent over the network as well.

Instead, a unique identification number for every email is generated locally and send to other agents if the corresponding email is spam. Therefore, it is impossible to conclude from this identification number (hash value) to the content of the corresponding email. Many procedures have been developed to deal with this issue. One popular solution is to hash the content of the email into a SHA-digest (secure hash algorithm) [9]. Although the SHA-digests which were created through hashing are easily to compare, they have a serious disadvantage. They cannot deal with a new form of spam. The content of this new form of spam is varied slightly from user to user, i.e. they contain details like the address of the receiver. So, although the email is from one sender and basically identical, the content of the spam message is slightly different for every user who receives it and hence, the SHA-algorithm generates a different digest for each message. Therefore, the SHA-algorithm cannot be used to compare the message contents. Our identification number generated for the content of emails follows a different idea.
For a certain amount of representative letters of the alphabet, the relative frequency of their occurrence in the content of the email is recorded. We line up the frequencies of these letters to a hash value. If we want to compare two hash values, the difference of the frequencies for each letter are formed and added up. The greater this value is, the more different are the frequencies of the letters and the more different are the two corresponding emails. This kind of identification number has several advantages:

- Emails with slightly different content are recognized as similar and spam email created in this way can be identified.
- Hash values of messages with the same content are equal to each other, their difference is zero.
- The hash value does not disclose any information on the contents of the emails to other users.

3.2 Spam Classification Procedure of the Antispam Agent

The principal unit of the spam filter network is the antispam agent. Every agent is assigned to a user. The agent is situated in the FIPA-OS platform treated in Section 2.2. The platform provides the antispam agent with a list of all other active agents in the network. This service is done by the Directory Facilitator. The agent stores the list of active agents in his own database, allowing the antispam agent to start communication with the other antispam agents. It sends them a request for new information about spam which the other agents have collected in the time where it was not connected to the network. Now, the other agent itself adds the sending agent to its database of active agents. The exchange of information is provided by the 'inform' communication act, part of the FIPA communication standard.

At the beginning of the classification process (cf. Fig. 2), the antispam agent downloads new messages from the mailserver of the user. The connection to a POP3 or IMAP server can be established using the Java Email Interface [4]. The

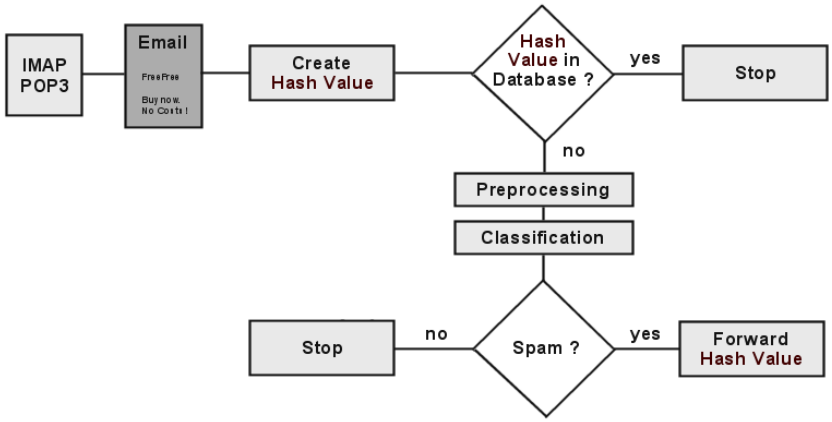

Fig. 2. Functionality of the antispam agent

configuration for the individual email account can be specified and saved by the user. After downloading and optionally deleting the email from the server, the antispam agent generates hash values for each of the downloaded emails. The process of generating hash values is explained in more detail in Section 3.1. These values are compared with the hash value entries of spam in the local database of the agent. Note that the database only contains hash values of messages which where clearly classified as spam by the agent itself or by another agent which sent this spam hash value through the network. If the hash value of the new email matches with one in the database, the email is spam. In case the content of the two compared emails only differs slightly, the difference of the two belonging hash values is rather small. The more different the content of two emails is, the greater the difference between the hash values is. The user can specify a threshold value. Emails whose similarity to each other lies over this threshold are regarded as similar. With the help of this similarity measure, spam emails from the same sender which only differ in a few letters (e.g. personal address of the receiver) are recognized as equal and are filtered. The spam email is transferred to a specific folder for later review (optionally the email can be deleted directly). If the hash value is not sufficiently similar to one of the spam hash values, it has passed the first filter. It is forwarded to the classification algorithm. Before the SVM algorithm can start to classify, the text of the email has to be preprocessed. The necessary preprocessing steps are as follows:

- *Removing HTML-Tags*
  Because the Email will be classified with the help of word occurrences, all other parts of the email must be deleted. There is also the possibility that the tags contain (for the user invisible) information to mislead the classification algorithm. Therefore all HTML-Tags are deleted.

- *Creating the Bag of Words*
  The words of the content of the email without HTML-Tags are collected into a list, also called the bag of words.
- *Stemming*
  The content of the bag of words is processed by the Porter Stemmer [8]. The amount of words to be considered for classification is reduced by stripping all suffixes of words to receive their root forms, i.e. stripping the words 'introduction', 'introduced' to 'introduc'. The meanings of the words remain, but the number of words to be considered is reduced.
- *Using Stop List (optionally)*
  To reduce the amount of words further, general words without special meaning which often appear in texts can be cut off, i.e. words like 'the', 'of', 'on' etc. This method is not undisputed, some authors explain that using the stop list reduces the accuracy of the classification algorithm [2].
- *Representing Text as Attribute Vectors*
  The remaining words are mapped into an attribute vector. If the attribute (word) is in the bag of words, its corresponding attribute receives the value 1 independently how often the word occurs in the text. Otherwise this attribute gets the value 0. The created vector is called 'binary vector', because it only consists of $values \in \{0, 1\}$. It is used in many text classification algorithms providing good classification results.

After the preprocessing, the actual classification is taken over by the support vector machines algorithm as a second filter. Before this algorithm can be applied, its classifier has to be trained based on training data. Training data consists of labelled email vectors, which have already been classified. The SVM algorithm builds up a training matrix from those classified vectors. Hereafter, the algorithm is ready to classify the unlabelled vectors of new emails. The preprocessed binary vector of the email is arranged either into the class 'spam' or the class 'nonspam'. The vector is classified using the matrix generated of training vectors. After the algorithm has arranged the email, the user has the possibility to change the classification judgement made by the SVM algorithm. Especially if the training matrix of vectors is small, there is a risk of misclassification. To increase the classification accuracy, the classified vector is taken into the local training matrix of the agent. With the growth of the matrix, the SVM algorithm quickly reaches a precision of over 95 percent [11]. Only if the email is classified as spam, its hash value is transferred to all other connected agents, which store this hash value in their local database. If they receive the same spam message, the recognize it at once, because the hash values generated from the newly received email is equal to the one received by the other antispam agent. Because it is impossible to infer the original content of the corresponding email from the hash value, the data transfer within the spam filter network preserves privacy, an important design objective. We explain why we do not send the classified email vectors through the network to antispam agents with a small amount of training vectors. This would power up their ability to classify emails correctly after a short time inside the spam filter network. But there are several big disadvantages

which stand against sending vectors over the network. First of all, there is the possibility to recreate the content of the emails by analyzing the corresponding vector. This security gap does not appear when sending hash values which where created to guarantee security. Furthermore, malicious users could try to reduce the quality of the network by sending thousands of misleading vectors to other antispam agents.

3.3 Interface of the Antispam Agent

Figure 3 shows the user interface to the database of hash values from an antispam agent which is connected to the network. The hash value generated from the current email is displayed in the box below the list. This value is compared to all values of the database. The list of hash values is regularly updated by exchanging spam information with other agents. The similarity threshold for emails can be adjusted with the slider on the right from 70 % up to 100 %. The tab 'All Email' leads to the list of emails loaded from the server. The tab 'Classification' shows the result of the SVM algorithm in classifying the current email. The other agents connected to the network can be seen under the tab 'Agents'. This interface allows to easy control the classification activity of the user's antispam agent. This is important for the acceptance of such a tool.

4 Conclusions

In order to deal with the huge amount of spam received day by day, powerful email filters with high reliability are needed. We combined multiagent systems with text categorization to identify spam. Information about spam emails spreads over the network to all other agents. If one agent has classified a certain spam email, all other agents profit from the hash value sent to them. They recognize the same spam email with absolute reliability. Similar emails (with different personal address for example) are also detected and recognized. By the data exchange, agents which enter the network are quickly provided with actual information about spam. Newly created agents receive hash values from other agents of the network and set up quickly an up-to-date database with spam information. The second filter, support vector machines, greatly supports the antispam agents at the task of classification of incoming unknown email in order to create new hash values. Every user can influence the creation of his individual training matrix by changing the classification results of SVM. Thus, he can specify the email to be classified as spam to its own taste. A lot of advantages arise from the use of the MAS platform FIPA-OS. Agents can register and deregister with FIPA-OS on the fly without influencing the stability of the system. FIPA-OS enables the communication between antispam agents and allows simultaneous and asynchronous email classification. This form of spam filtering is attractive for both the normal user as well as enterprises.

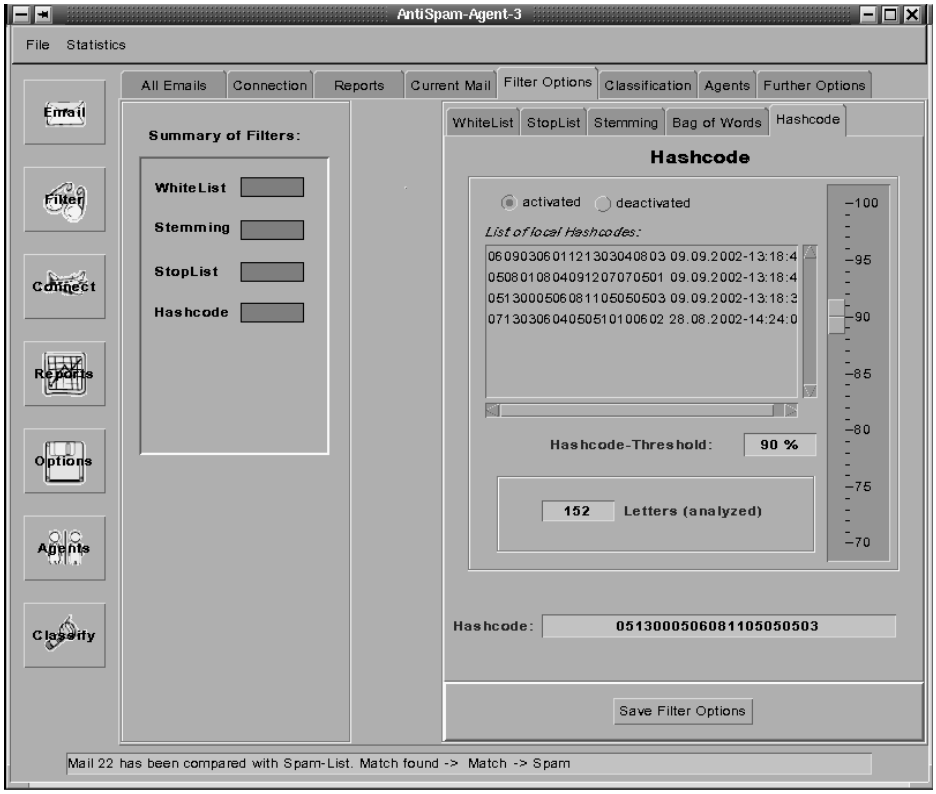

Fig. 3. Screenshot of the user interface of the antispam agent

5 Future Work

There are still two disadvantages within our spam filter approach. Firstly, a huge amount of data overhead is produced by the regularly exchange of hash values. Furthermore, it is possible for malicious users to flood our network with thousands of false hash values. In order to eliminate these drawbacks, we are currently working on a simulation platform of a distributed spam filter network with up to 100 agents where antispam agents will evaluate the hash values they receive and establish trust to agents which provide useful and correct hash values to identify spam. After exceeding certain thresholds, benevolent agents that receive the same spam mails will group together, exchange hash values within their group and use only hash values provided by trusted group members to identify spam. In this way, we will eliminate the two disadvantages mentioned above. The amount of data which is transferred over the network will be reduced by forcing the hash value exchange within the group of agents which receive the same spam. Malicious agents will be excluded from the groups of benevolent agents and will be prevented from flooding these groups with false hash values.

Within the scope of the simulation platform, we will evaluate if the exchange of hash values within the network of antispam agents increases the total number of spam mails which are identified correctly.

With a few changes, our spam filter could also be used to assign emails to certain categories of interest. A user could sort emails about 'sports' or 'news' in the corresponding folders. Furthermore the filter network can be enhanced with several other components in the future. A whitelist filter which specifies a list of trusted email senders has already been included. Email from these people will never be classified as spam. Further filters can be implemented: Blacklists could be set up with addresses of people and organizations which sent spam email in the past. Their email could be blocked by the filter. The modular structure of the antispam agents with all modules written in Java provides space for additional identification methods for spam.

References

1. W.W. Cohen (1996): Learning Rules that classify e-mail. Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access 18–25, AAAI Press.
2. V. Vapnik, H. Drucker, D. Wu (1999): Support Vector Machines for Spam Categorization. In: IEEE Transactions on Neural Networks, 10(5):1048–1054.
3. Homepage of the Foundation for Intelligent Physical Agent (FIPA). http://www.fipa.org
4. Java Mail (TM) API Design Specification. Sun Microsystems, Inc.: http://java.sun.com/products/javamail/JavaMail-1.2.pdf
5. S. Shankar, G. Karypis (2000): A Feature Weight Adjustment Algorithm for Document Categorization. Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD 2000).
6. P. Pantel and D. Lin (1998): SpamCop - A Spam Classification and Organization Program. Proceedings of AAAI-98 Workshop on Learning for Text Categorization 95–98, AAAI Press.
7. Nortel Networks Corporation FIPA-OS Informations: http://www.nortelnetworks.com/products/announcements/fipa
8. M.F. Porter (1980): An algorithm for suffix stripping. In: Program, 14(3):130–137.
9. Secure Hash Standard (1995): Federal Information Processing Standards Publication 180-1. National Institute of Standards and Technology.
10. Spam Statistic: http://www.raingod.com/angus/Computing/Internet/Spam/Statistics/index.html
11. V.N. Vapnik, D. Wu (1998): Support Vector Machine for Text Categorization. AT&T Research Labs, http://citeseer.nj.nec.com/347263.htm
12. Vipul's Razor Homepage. http://razor.sourceforge.net
13. J.D.M. Rennie (2000): ifile: An Application of Machine Learning to E-Mail Filtering. Proceedings of the KDD-2000 Workshop on Text Mining, Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
14. M. Wooldridge, N. Jennings (1995): Intelligent Agents: Theory and Practice. In: Knowledge Engineering Review 10(2):115–152, Cambridge University Press.