# A Trainable Fuzzy Spam Detection System

**M. Muztaba Fuad**
Dept. of Computer Science
Montana State University
Bozeman, Montana, USA
fuad@cs.montana.edu

**Debzani Deb**
Dept. of Computer Science
Montana State University
Bozeman, Montana, USA
debzani@cs.montana.edu

**M. Shahriar Hossain**
Dept. of CSE, Shahjalal University
of Science & Technology
Sylhet-3114, Bangladesh
shahriar-cse@sust.edu

## Abstract

*Electronic mail (e-mail) has been considered as one of the most convenient way to communicate among the users in the Internet. The rapid growth of users in the Internet and the abuse of e-mail by unsolicited users cause an exponential increase of e-mails in user mailboxes. Although there are several systems which use different AI techniques to filter out spam, there is hardly any system developed so far to filter e-mails using the fuzzy logic system. This paper presents the design and implementation of a trainable fuzzy logic based e-mail classification system that learns the most effective fuzzy rules during the training phase and then applies the fuzzy control model to classify unseen messages. Our findings imply that automatically trainable fuzzy spam filters are practically viable and can have a significant effect on spam detection.*

## Keywords

E-mail, SPAM, HAM, Fuzzy logic, Machine learning.

## INTRODUCTION

With the rapid extension of the Internet, e-mail has been considered as one of the most efficient and convenient way of communication. However, recently the increasing popularity and low cost of sending an e-mail make it very attractive to the direct marketers. It is now become very easy to send unsolicited messages blindly to thousands of people at no cost at all by using easily available bulk-mailing software and large lists of email addresses typically harvested, even purchased or rented [3] from web pages and newsgroup archives. Therefore, the volume of this unsolicited bulk e-mail or spam that shows up in the user's mailbox daily has been increasing exponentially.

Spam messages are nuisance and huge problem to most users since they clutter their mailboxes and waste their time to delete all the junk mails before reading the legitimate ones. They also cost user money with dial up connections; waste network bandwidth and disk space and most importantly make available harmful and offensive materials such as pornographic sites to children. A survey in year 2002 estimates that the amount of spam has increased 600% in that year [9], another estimates that 12-15% of all email traffic is spam [10]. Although many governments legislate against spammer, there is very limited effect so far. Therefore, we need to investigate technically reliable and efficient solution to combat the battle.

The most popular and direct way to prevent spam is the anti-spam filters, software tools that block spam messages automatically. These anti spam filters vary in functionality from blacklist (frequent spammer list) and whitelist (trusted user list) to content-based filters. The latter is more powerful since spammers generally use false addresses. Existing content-based filters can be categorized as rule-based, key-word based and learning based. The keyword based filter [11] utilizes a dictionary of common spam phrases and search for a particular pattern in the messages. While they perform well, they need to be maintained and tuned constantly since the characteristics of spam messages change over time. Rule-based filters [1, 16] generally use a wide range of tests to recognize spam features and assign a 'spam score' to every email. Although some of them are still popular, they also require periodic update and maintenance. As classification rules are often fixed and since the classification of 'good' and 'bad' spam often differs from person to person, fixed classification methods are unlikely to provide good performance for all users. The last category of anti spam filters are still emerging and automatically learn how to block spam messages by processing previously received spam and legitimate messages.

Sahami et. al. [15] first proposed a machine learning algorithm to construct a filter that can learn automatically. They trained a Naïve Bayesian classifier on previously categorized spam and legitimate (also called ham) messages and came up with an inspiring performance on unseen messages. After their success, a range of machine learning techniques (Decision tree [8], support vector machine [6], k nearest neighbour [5], boosting [1]) have been used for spam detection with promising results. In recent years researchers are increasingly using machine learning techniques to automatically build anti-spam filters.

In this paper, we use a trainable Fuzzy classifier [13] to build an automatic anti-spam filter. Trainable fuzzy system is a fuzzy logic based system that derives the (fuzzy) classification from training data using learning techniques. The motivation of using fuzzy logic for spam detection came from the fact that there is no clear separation between spam and non-spam messages and fuzzy logic is a good way to deal with those fuzzy boundaries. Fuzzy classification assumes the boundary between two neighbouring classes as a continuous and

overlapping area within which an object has partial membership in each class. This viewpoint not only reflects the reality of many applications in which categories have fuzzy boundaries like spam detection, but also provides a simple representation of the potentially complex partition of the feature space. While fuzzy logic can reason with imprecise information, they can not acquire knowledge automatically. To overcome this problem of knowledge acquisition, a learning algorithm is used in this system to automatically extract most efficient fuzzy rules during training. For each fuzzy rule a 'grade of certainty' is computed which is adjusted during the learning procedure. The output of each rule is then weighted by the grade of certainty during the classification process.

The approach used in this paper uses different fuzzy sets to encode various features extracted from an email message and according to some fuzzy rules classify the message as spam or legitimate. Feature extraction not only detects 'spam word' and 'spam phrase', also identifies other aspects of an email such as an empty From/To field which are strong indication of a message being a spam. Since many anti-spam filters [7, 15] compare the number of 'spam' words in a message to the number of 'non-spam' words to calculate a percentage, spammers now a days use a special technique [3] which includes random legitimate text inside a message in order to offset this percentage to a degree where the message is accepted as a deliverable. To overcome such problem, our system extracts a wide range of features from a message and is therefore hard to fool. We evaluate the system on a self constructed corpus and the results show that trainable fuzzy spam detection system performs reasonably well with 80% accuracy on detecting spam. The system is also more adaptive since it can always learn from the training examples and fine tune itself based on new information. It is also easy to update and maintain and most importantly its behaviour is comprehensible.

The remaining of this paper is organized as follows: section 2 describes the overall system architecture and discusses about various design and implementation issues, section 3 presents the result and section 4 provides the conclusions and future works.

## DESIGN AND IMPLEMENTATION

The overall design of the system is shown in Figure 1. At first the messages from the corpus is parsed and important features are extracted. Then the feature values are passed to the respective fuzzy sets for fuzzification. Based on the fuzzified input signals, a number of fuzzy rules are triggered in parallel with various values of firing strength. The rule outcomes are then aggregated and defuzzified and based on the output a prediction is done. We now turn into the detail analysis of each component of the system.

### Corpus

As the main objective of this study is to justify the use of fuzzy logic in spam detection, for simplicity we limit the scope of our study to emails that contain plain text only. This design decision restricts us from using the publicly available spam corpuses directly. Instead, the plain text spam messages used in this study are obtained from a spam corpus [18] by excluding all emails organized as HTML. The legitimate messages used are the messages received by the authors for four months. The final corpus constructed in this way contains 301 messages all together, 144 of them are spam and 157 of them are legitimate messages.

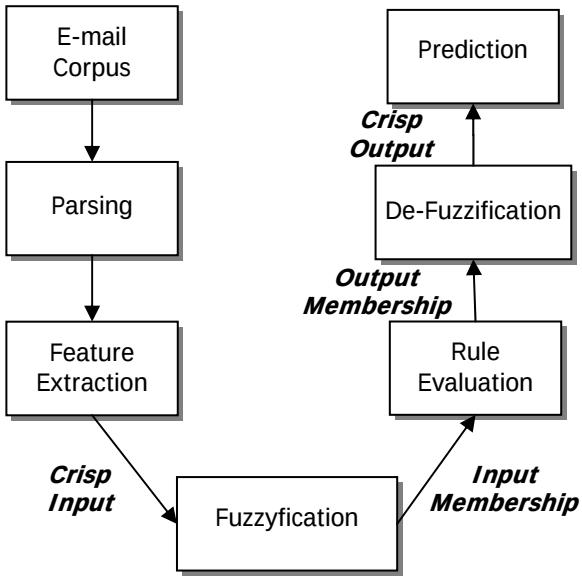


**Figure 1.** Overall system architecture.

### Feature Extraction

Spam generally contains many distinctive features all over the email. Referring to RFC 822 [14], an email usually consists of two main parts, which are header and body. The header of an e-mail provides some tagged information, while the body is usually unstructured text. Our spam detector extracts the fields From, To, Cc, and Subject from the header of an email. There are much useful information available in those fields, such as missing or malformed From field, Subject containing spam phrases, etc., which has great importance as a feature to detect spam. For flexibility we also associate a weight to each feature.

The Body part of an email is also analyzed to extract several important spam features. The system takes into account that, most spam either includes the URL of a site they want the customers to visit or contain a toll free number they want the customers to call and therefore the system extracts those features. Most spammers also use characters like $, #, ! repeatedly in their email. The system identifies those characters in the email, count their numbers and use the resulting number as a feature. The field Subject and Body are also searched for predefined Spamphrases and Spamwords. To get the words of the Spamword list we first tokenize every word of all spam and legitimate messages of the corpus used and create two

separate lists of each category containing occurring words and their frequencies. During tokenization, we ignore tokens that are all digits and case folding [12] is used to disambiguate between two words with the same meaning. Then we exclude the words that are common in both list (in that way most stop words [12] are also removed) and then finally we merge the spam and legitimate words in a single list Spamword and assign weight to each word based on the number of times they appear in spam or ham category of emails. Words that appear in non spam messages are assigned a negative weight respective with their frequencies.

### Trainable Fuzzy Classifier

A fuzzy classification system consists of a rule base which contains a set of fuzzy classification rules and an inference engine which maps a given input to an output using that rule base. We followed the Mamdani [4] fuzzy inference model in this implementation because of its simplicity. In total, five input fuzzy sets are used: two for features extracted from the header part of an email and the rest three for the body features. All of the input fuzzy sets are trapezoidal and are represented by linguistic terms: Low, Mid, High which are then characterized by appropriate membership functions. The output fuzzy set contains only two linguistic terms: Spam and Non-Spam. After extracting the features, the respective crisp inputs for each fuzzy set are passed to the fuzzification layer. The fuzzification stage then determines the degree to which this input belongs to the respective fuzzy set. A fuzzy rule receives inputs from the fuzzification layer that represent fuzzy sets in the rule antecedents. A typical fuzzy classification rule of our system has the form:

*IF X1 is Low AND X2 is Mid AND X3 is High AND X4 is High AND X5 is High THEN Y is Spam*

Here X1 to X5 are crisp inputs representing feature values. The conjunction of the rule antecedents is evaluated by fuzzy AND operation. The consequents of different triggered fuzzy rules with various values of firing strengths are then combined using the fuzzy OR operation. That value is used as an input for defuzzification, which employs a standard centroid [4] technique and produces a crisp output. We then compare the crisp output to a predefined threshold value and predict the message as Spam or Non-Spam.

To learn from the training messages, we use a learning algorithm that learns the importance or reliability of a rule during the training process. At first, the system automatically generates a complete set of fuzzy IF-THEN rules. Our system consists of 5 fuzzy sets (with 3, 3, 2, 3, 3 linguistic values or states) and therefore consists 324 rules all together. For the same combination of antecedents, the rule base contains two different rules with consequent Spam and Non-Spam. Initially all rules are set to a have a 'grade of certainty' or weight value of 0.5. For each training example, this weight is adjusted as follows:

When the training example is correctly classified by $Rule_i$, adjust Wi by:

$$W_i = W_i + \alpha \times (1 - W_i)$$

Conversely, when the training example is not correctly classified, adjust Wi by:

$$W_i = W_i - \beta \times W_i$$

Where α and β is learning constant and α < β

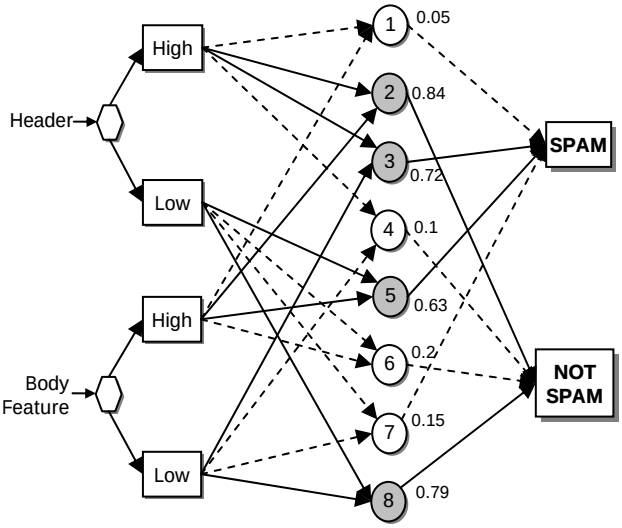


**Figure 2.** Rule extraction.

The training continues until the classification error on the training examples is less than 0.1. Once the training is done, the rule containing the greater weight between the two rules having the same antecedents has been selected as correct. And thus the final rule base contains 162 correct rules and can be used for the classification of unseen messages. The simple example of Figure 2 (with two input fuzzy set with linguistic value Low and High) demonstrates how the correct rules are extracted. At the end of training, correct fuzzy rules (2, 3, 5, 8) are extracted and bad rules are discarded (1, 4, 6, 7) from the system.

## EXPERIMENTS AND RESULTS

We use a ten-fold cross validation method to evaluate the system. Ten fold cross validation has been proven to be statistically good enough in evaluating the performance of the classifier and in reducing random error [19]. The corpus is equally divided into ten different subsets. Nine out of ten of those subsets are used to train the classifier and the tenth subset is used as the test set. The procedure is repeated ten times, with a different subset being used as the test set. Results are then averaged over the ten runs. The resultant learning curve of the system is shown in Figure 3.

We also constructed a confusion matrix (contingency table) to evaluate the classifier's performance. Table 1 shows a generic contingency table for a binary class problem. True positives (TP) denote the correct

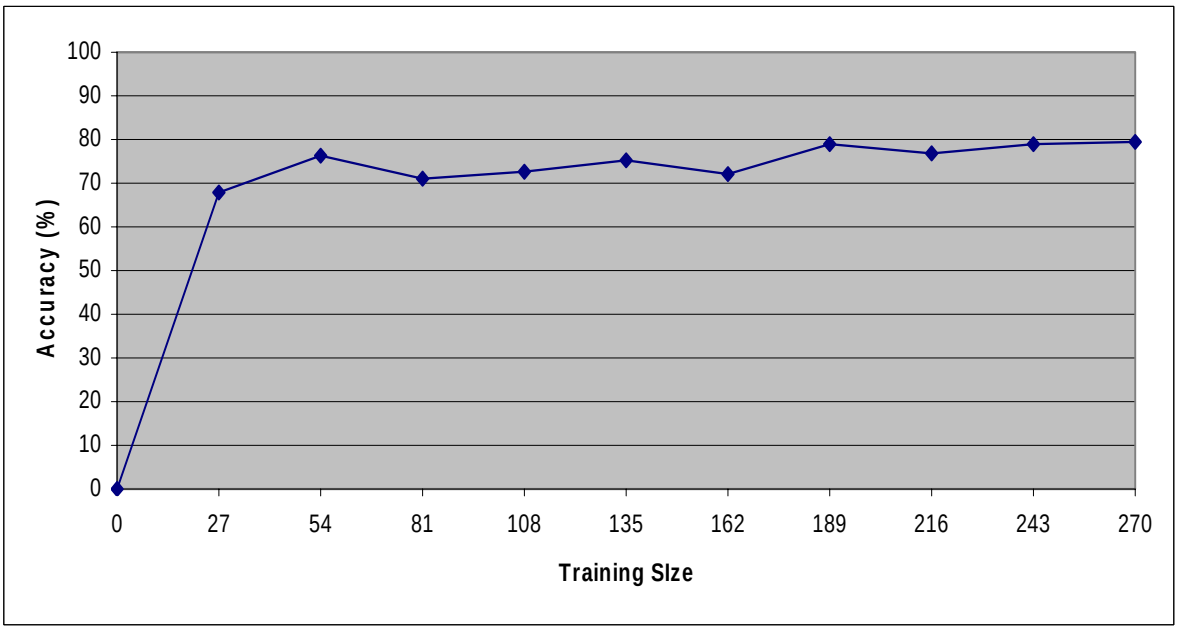


**Figure 3.** Learning Curve.

classifications of positive (spam) examples. True negatives (TN) are the correct classifications of negative (ham) examples. False positives (FP) represent the incorrect classifications of ham examples into class spam and False negative (FN) are the spam examples incorrectly classified into class ham.

**Table 1**: Contingency table for binary class problem.

| | | Predicted | |
|---|---|---|---|
| | | Positive | Negative |
| Actual | Positive | TP | FN |
| | Negative | FP | TN |

Based on the contingency table, several measurements can be carried out to evaluate the performance of the classifier. The most popular performance evaluation measure used in prediction or classification learning is classifier accuracy ($A_{cc}$), which measures the proportion of correctly classified instances:

$$A_{cc} = \frac{TP + TN}{TP + TN + FP + FN}$$

Spam precision denotes the percentage of messages in the test data classified as spam which truly are and spam recall denotes the proportion of actual spam messages in the test set that are categorized as spam by the classifier. Spam precision ($S_p$) and spam recall ($S_r$) can be defined as follows:

$$S_P = \frac{TP}{TP + FP}$$

$$S_r = \frac{TP}{TP + FN}$$

The results are shown in table 2. In spam filtering, people are generally more concerned about mistakenly blocking a legitimate message than letting a spam message pass the filter. Therefore spam precision is the most important factor to most users and from the results, we can see that the proposed system has the potentiality to fulfil this concern.

**Table 2**: Results.

| Total Msg. | Train Msg. | Test Msg. | Spam Accuracy | Spam Precision | Spam Recall |
|---|---|---|---|---|---|
| 301 | 271 | 30 | 90% | 83% | 72% |

## CONCLUSIONS

In examining the growing problem of dealing with spam messages, we found that it is possible to train a fuzzy spam filter to automatically learn effective fuzzy rules so that a large portion of such spam messages can be eliminated from a user's mail box. The efficacy of such filters can also be greatly enhanced by considering not only the full text of the email messages to be filtered, but also by extracting a set of features of great importance from other parts of the message. The use of fuzzy model allows us to integrate domain specific expert knowledge to the learning task and make the system more adaptive. Our results show that the fuzzy model with optimized rule base performs reasonably well. We interpret the result as implying that, automatically trainable fuzzy spam filters are practically viable and can have a significant effect on spam detection.

In future work, we like to filter e-mail messages having HTML in its body. Since we now have a filtering mechanism, additional component that can parse HTML messages can be plugged in very easily. Once we

incorporate HTML messages, we can use large spam corpus to train our system, which will certainly increase the accuracy. Then we can also handle most of the e-mail obfuscation techniques [3]. We are also interested in extending the feature extraction module so that non-spam features such as message containing signature or message coming from a trusted list of users, can also be extracted. Finally we like to extend our fuzzy model so that membership functions and other parameters of the fuzzy sets can also be learned automatically.

## REFERENCES


[1] Carreras, X. and Mdrquez, L., "Boosting trees for anti-spare email filtering", In Proc. of RANLP, 2001.

[2] Cohen, W.W., "Learning Rules that Classify E-Mail.", Proceedings. of the AAAI Spring Symposium on Machine Learning in Information Access, Stanford, California,1996.

[3] Cournane, A. and Hunt, R., "An Analysis of the Tools Used For the Generation and Prevention of Spam", Computer and Security, Vol. 23, pp 154-166, 2004.

[4] Cox, E., "The Fuzzy System Handbook", Academic Press, Second Edition, 1999.

[5] Daelemans, W., Z. Jakub, K. van der Sloot and A. van den Bosch, TiMBL: Tilburg Memory Based Learner, version 2.0, Reference Guide. ILK,Computational Linguistics, Tilburg University. http:/ilk.kub.nl/~ilk/papers/ilk9901.ps.gz, 1999.

[6] Drucker, H., Wu, D., & Vapnik, V., Support vector machines for Spam categorization. IEEE-NN, Vol. 10, No.5, pp. 1048–1054,1999.

[7] Graham, P., Better Baysian Filtering. In Proceedings of Spam Conference http://spamconference.org/proceedings2003.html, 2003.

[8] Hidalgo, J. G., Spez, M, and Sanz, E, Combining text and heuristicz for cost-sensitive spam filtering. In Proc. of CONL, 2000.

[9] Lee, J., "Spam: An escalating attack of the clones", The New York Times, 2002.

[10] Mayer, C., and Eunjung-Cha, A., "Making spam go splat: Sick of unsolicited e-mail, businesses are fighting back", The Washington Post, 2002.

[11] Microsoft Outlook, www.microsoft.com, 1998.

[12] Norvig P. and Russell S., "Artificial Intelligence A Modern Approach", Prentice Hall, New Jersey, 2003.

[13] Nozaki, K., Ishibushi, H. and Tanaka, H., "Trainable Fuzzy classification systems based on Fuzzy If-Then-Rules," Proc. IEEE, vol. 1, pp. 498-502, 1994.

[14] RFC 822: Standard for the Format of Arpa Internet Text Messages, www.w3.org/Protocols/rfc822/, 1994.

[15] Sahami, M., Dumais, S., Heckerman, D. and Horvitz, E., "A Bayesian Approach to Filtering Junk E-Mail. In Learning for Text Categorization", AAA1 Workshop, pp. 55-62, Madison Wisconsin, 1998.

[16] SpamAssassin, www.spamassassin.org, 2004.

[17] SpamButcher, www.spambutcher.com, 2004.

[18] The Great Spam Archive, www.annexia.org/spam/, 2004.

[19] Witten, I., H. and Frank, E., "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", Morgan Kaufmann Publishers, San Francisco, CA, 2000.