Exploration of Neuro-Fuzzy Spam Filtering based on Naive Bayesian Filters

Madhujit Ghosh and Jonathan J. Guernsey
Dr. Devinder Kaur
EECS 6360 - Knowledge Based Systems

Abstract: A text parser was used to calculate the statistical distribution of words within an email body. This information was used by a neuro-fuzzy system to determine the spam classification of the email. This process of detecting spam in an email was experimentally found to be 90% efficient. This design is exceptionally good as compared to present day filters based on its simplicity and limited scope of detection methods. Our system could be further improved by incorporating other identifiers of email spam.

Introduction

Spam is unsolicited email on the internet. A major problem facing internet computer users today is the deluge of unwanted and often rude email waiting in their email-boxes every morning. This unsolicited email is flooding the Internet with many copies of the same message, in an attempt to force the message on people who would not otherwise choose to receive it. Most spam is commercial advertising, often for dubious products, get-rich-quick schemes, or quasi-legal services. Spam costs the sender very little to send. Most of the cost is absorbed by the recipient or by the carriers rather than by the sender. This is mostly in the form of lost productivity or network resources.[5] Current estimates suggest that up to 30% of today's e-mail volume is spam, a figure likely to increase over time. In addition to the unnecessary strain it places on a corporate network, spam frequently contains viruses.

According to Paul Graham in his now well known internet article, "A Plan for Spam."

> … they (spammers) have to deliver their message, whatever it is. If we can write software that recognizes their messages, there is no way they can get around that."

Paul Graham introduced the idea of using Naive Bayesian Filters (as pure discrete sets) in his article.[3] This information was further assimilated by I. Androutsopoulos et al in their article "An Evaluation of Naive Bayesian Anti-Spam Filtering." [1] We have used this idea of machine learning and applied fuzzification and artificial neural network techniques to further simplify and yet increase the efficiency of spam classification.

The project development was divided into three separate parts. We wrote a word filter that sifted through the text contained inside the body of an email, parsing it using (slightly modified) Naive Bayesian methods and normalizing weighted values producing a fuzzy output. A fuzzy algorithm and classification scheme was developed for the text parser. The next step was to develop a neural network which could be trained with the fuzzy classifiers from the previous section of the project. When the fuzzy classifiers generated by the text parser were passed into the trained neural network, it outputs a spam/no spam identifier for the input email. The network was trained with a number of spam and non-spam emails before testing was performed with previously unseen data.

Text Parser Design

The whole project, including the neural network was designed and written in Visual C++ 6.0 SP5. The code is attached at the end of this report. The GUI for the project was written using Microsoft Foundation Classes (MFC).

The parser takes an input as a simple text file from the user. Using the GUI, the user browses their computer and chooses a text file which is to be classified for spam. To simplify the design aspect of the project, we copied and pasted the emails from an email client to Notepad and saved them as simple text files. The parser reads in the text file and counts the number of times each word appears in the email and stores them in a dynamic array. This array is sorted when the end of the text file is reached. This dynamic array is then passed to the second stage inside the parser, the fuzzification algorithm.

The array containing the parsed words is fed into the fuzzification algorithm. Two separate text files store the good and bad fuzzy identifier values for well known and commonly found words in non-spam and spam emails respectively. The fuzzy identifier for each word is weighted according to the importance of the word in association with spam. Words which could immediately imply that an email is spam have the highest weights, while others, which could be present in good emails are provided with lower weights. The word lists were maintained as text files so that they could easily be added to by the user. The words and their associated weights are read into separate data trees. The next part of the program uses simple word matching techniques. When it comes upon a word in the main body of the email which is present in either of the word lists, it multiplies the associated weight of the word from the list with the number of times the word showed up in the email and keeps track of these values.

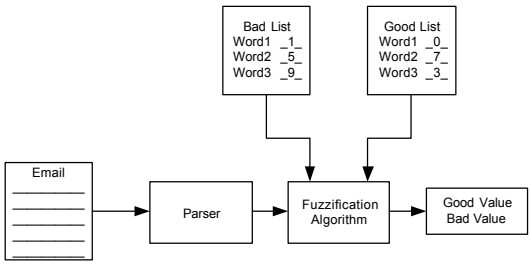

Figure 2.1: Graphical representation of text parser (fuzzification algorithm)

Once the total weights of the good and the bad words have been determined by the program, we used two separate methods of generating an output. Both of these methods were implemented into the program and tested to determine the best method of classification. The first method takes a minimum of the good weighted values and a minimum of the weighted values of the good and bad words in the email. These weights are then passed into the neural network to be further analyzed. The equations defining this method are shown in equation 2.1.

$$\min(w_{x,i} g_{x,i})$$

x = good or bad words
$?_x$ = number of occurances of good/bad word
w_x = weight of $?_x$

Equation 2.1

The second method follows the same steps of word comparison between the statistical distribution and the good/bad word lists, but it computes the sum of the good weighted values and the sum of the bad weighted values and divides them by their total number of occurrences, thus normalizing the output of the system of word classifiers. This output is then passed onto the neural network to be used to either learn from or classify according to the network rules. The equations defining this method are shown in equation 2.2.

$$\frac{\sum_{i=1}^{n_x} w_{x,i} g_{x,i}}{n_x}$$

x = good or bad words
$?_x$ = number of occurances of good/bad word
w_x = weight of $?_x$

Equation 2.2

Artificial Neural Network Design

An artificial neural network is a collection of mathematical models that emulate some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning.[2] Of most interest is the emulation of the decision making process as it applies to object recognition. As an example, a normal person can recognize that a campfire is dangerous by simply recognizing the flames as fire. A person is normally taught that fire is "hot", "will burn", or in some cases the person may have learned from experience by being burnt. In all cases this training or experience has taught the person that fire is dangerous and painful. This training has been trained into the persons brain or "biological neural network" When the person comes upon the campfire and views the flame, they will recognize the presence of heat as well as the shape of the flame among other attributes and will almost immediately recognize that it is fire. This will link the person to the fact that it is dangerous and will attempt to avoid contact with it. This concept of classification of an object can be modeled using a neural network. Using the input from sensors of a sort, information is applied to the inputs of the neural network to be processed to achieve the output classification. In the case described, the heat is detected by the skin, visual shape of the flames by the eyes, each of these inputs are

applied to the brain's neural network along with other attributes and the classification of fire is determined. It is this classification of information as well as the fact that artificial neural networks can be trained that makes them readily applicable to an email filter. Since the email filter is a classifier that should recognize an email as either spam or not-spam, a neural network should be able to take in information about the content of the email and be able to learn and classify them. To implement a neural network for a simple two pattern classification, the most simplistic neural network was used, a single neuron. A single neuron has the ability to make a single classification between two patterns, although it has the limitation that the two classes must be linearly separable. A linearly separable classification is defined as classification between two patterns that can be accomplished using a single line. As shown in Figure 3.1, the logical AND function is a linearly separable classification while the logical XOR (exclusive-or) function is not linearly separable.

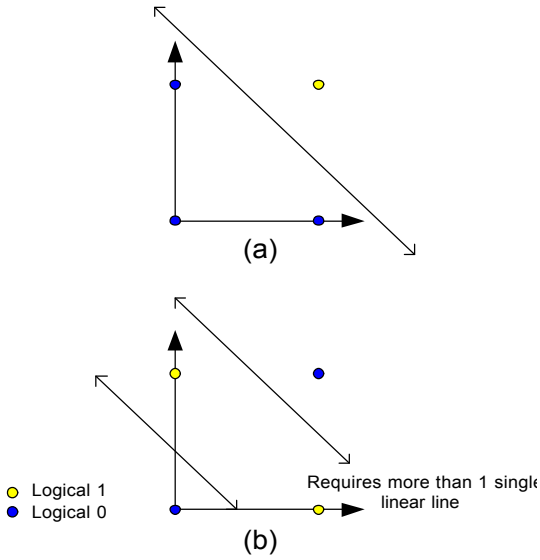

Figure 3.1 (a) Logical AND Function (b) Logical XOR Function

To simplify this initial exploration of the email filter, the assumption that spam and not-spam pattern classifications are linearly separable is imposed. Using this assumption allows us to simplify the neural network to the single neuron as well as simplify the neural network training that is discussed in section IV. Since the two fuzzified outputs of the fuzzy algorithm portion of the filter are applied to the input terminals of the artificial neural network and the output of the network is a spam or not-spam classification, a perceptron was chosen for the single neuron. The design of the perceptron is a standardized design as shown in Figure 3.2.

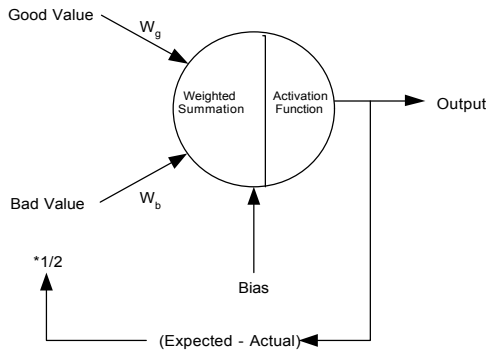

Figure 3.2 Perceptron

The perceptron summation function, f is a weighted sum as defined below. The activation function is the unipolar step function shown in Figure3.3.

$$f = \frac{\sum_{i=0}^{n} w_i x_i}{\sum_{i=0}^{n} w_i} - q_{bias}$$

x = input value of input i
w = weight value of input i

Equation 3.1

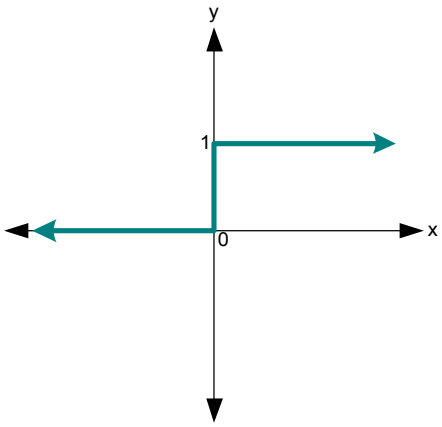

Figure 3.3 Unipolar Step Function

This perceptron works well with the current filter design because the perceptron is already designed in the correct format to work with the outputs of the fuzzification algorithm. Using the two weighted sums, good "words" and bad "words", the neural network attempts to ascertain the spam classification. This is accomplished by comparing the summation value against the linear separation that is defined by training the neuron. As an example, if the bad weighted value was 1000, while the good weighted value was 100, the email would likely be classified as a spam email, because the bad weighted value is greater than the good weighted value by a factor of 10. The neural network training tunes this comparison operation to make it more efficient in classifying the email.

Neural Network Training

The training of the neural network perceptron is done using a modified version of the single discrete perceptron training algorithm (SDPTA) using pattern training mode which updates the weights after each pattern is processed.[6] The training is only considered complete when the neural network can process all training patterns without a singular misclassification. This algorithm uses an expected value and back propagation to adjust the weights of the perceptron after each pattern (set of inputs) is processed within the neuron. The normal SDPTA re-weighting equation is shown below.

$$\Delta w = \frac{a}{2}(\text{expected} - \text{actual})^2$$

Equation 4.1

This equation was modified in our simplistic design for efficiency. The squared factor was removed from the adjustment equations for both the weights and bias values. The learning factor a/2 was also removed from the bias re-weighting equation giving the following equations.

$$\Delta w = \frac{a}{2}(\text{expected} - \text{actual})$$

Equation 4.2

$$\Delta bias = (\text{expected} - \text{actual})$$

Equation 4.3

The values of the weights and bias for the neuron are stored in a configuration file that is loaded during the startup of the filter and updated after training has been performed. These values allow us to continually add to our training set without re-training the neuron over information that it has already learned. It is assumed that the trainer is adding to the initial training set as opposed to writing an entirely new training set which would completely override all previous training. In our actual training the learning constant was set to the standard value of 1. This was done to decrease the rate of change for the re-weighting to allow for a more refined linear separation between the spam / not-spam classifications. The values of the initial training set for good weighted values and bad weighted values was designed to keep away from extreme differences in value. The training set uses small differences to allow the linear separation to classify close values which would occur neat the separation boundary.

Experimentation

Using the training file, we trained the system with 14 emails. There were 7 spam emails and 7 non-spam emails included in the test. To simulate randomness in the system as is found in real life, the good and bad emails were spread out over the training set. Each training email was followed by an expected spam classifier for that email. The trained network was then used to classify 200 different emails from different sources which had not previously been seen by the system. At the end of the checking process, we ended up with 90% correct answers and 0% false positives. A false positive would be a more serious error than a false negative as a false

positive would mean an important email being identified as spam and deleted. A false negative would denote a spam email that was classified as non-spam.

Results and Conclusions

The efficiency of email spam filters can vary upon the product and their placement within an email network. It is very difficult to generalize the efficiency of a group of products without looking at implementation. Mailfilter.com assures customers that their product will "reduce unwanted e-mail by 80 to 95 percent' while other companies tout a 95 to 100 percent success rate for eliminating spam.[4] However, all of these statistics are dependant on a number of salient factors. Very strict spam filters generate a huge number of false positives which increases their hit rates to close to a hundred percent. Even though these products look like they are very effective in spam detection, they result in the loss of important emails. Also, many of the filtering techniques used by commercial filters look at a variety of aspects of an email before classifying it. While comparing our results with the commercial spam filters available today, we have to keep in mind that our filter uses only one aspect of emails to identify it as spam. Keeping this simplicity of design in mind, a comparison between our results and that of industry filters leads us to believe that our method is very useful and quite effective in spam filtering. The next section talks extensively about the modifications that could be made to the system to increase its efficiency and make it more useful in real world applications.

Future Directions

This system is extremely capable of parsing through text emails and classifying them as spam or otherwise. However there are some limitations inherent in the system. These limitations were posed by the lack of time and scope of the project. Some very small modifications and module additions would make this system much more robust and versatile.

Most end users use email filters to parse through a small number of emails and try to increase productivity by not having to look through many unwanted emails. If this system were incorporated into an existing email client or was extended into a complete email client capable of receiving and sending emails, it would be a perfect addition to any desktop system. One future direction for this project would be to add email retrieval (POP) and sending (SMTP) functionality to the system.

At the present state of the program, it only looks at the text that is contained inside the body of the email. Another future improvement would be the addition of more modules into the neural network to look at other aspects of the email. The subject line of the email, the sender of the email and the images contained inside the email could be other important identifiers of spam. Another improvement to the neural network would be to design a more complex neural network model that could bypass the assumption that spam / non-spam classifications are linearly separable. More and more spam seems to contain links to different sites on the internet. If we could setup a way for the program to follow the links to these sites and parse the text contained in those sites, we would come up with even more powerful classification methods.

One feature of this program that we would like to mention here is that simplicity and versatility of the neural network design and the efficiency of the data structures used for the parsing problem. This program would function fine for a single user as a desktop spam filter. However, it would really shine as a high end server spam filtering system. The data structures used to parse email text is powerful enough to crunch through thousands of emails in very short amounts of time and this program would be a powerful addition to enterprise PROCMail or QMail server.

In conclusion, we have used a modified neuro-fuzzy model to effectively understand the simple word patterns that exist in unwanted and warranted emails. We have further supported our hypothesis with the design and construction of a program which accurately identifies these patterns in internet email, learns from the output of these patterns and uses them to classify new emails, thus helping to simplify a complicated and growing problem on the internet today.

References

[1] Androutsopoulos, Ion, John Koutsias, Konstantinos V. Chandrinos, George Paliouras and Constantine D. Spyropoulos. "An Evaluation of Naive Bayesian Anti-Spam Filtering" *Machine Learning in the New Information Age: Proceedings of the Eleventh European Conference on Machine Learning, Barcelona, Spain. 2000*. Eds. G. Potamias, V. Moustakis, and M. van Someren. Barcelona: 2000. 9-17.

[2] *Artificial Neural Networks Page.* 1997. Battelle Memorial Institute. 16 Dec. 2002 <http://www.emsl.pnl.gov:2080/proj/neuron/neural/what.html>.

[3] Graham, Paul. "A Plan for Spam." Spam Discussion and Research Site. 16 Dec. 2002 <http://www.paulgraham.com>.

[4] *Mail-Filters.Com: Premier Spam Filtering Solutions for Enterprises* . 2001-2002. San Mateo, CA. 16 Dec. 2002 <http://www.mail-filters.com>.

[5] Mueller, Scott Hazen. "Fight Spam on the Internet!" Spam Abuse Information Page. 16 Dec. 2002 <http://spam.abuse.net>.

[6] Zurada, Jacek M. *Introduction to Artificial Neural Systems*. West Group, Jan. 1992.