Research Proposal

Title: Email Spam Detection using a Multi-Objective Memetic Algorithm

Author: James Dudley

Supervisor: Dr. Luigi Barone

Background
Unsolicited bulk email, better known as spam, is an annoying reality of internet use today. Because the cost of ‘spamming’ is so low it is widely used, indeed, today spam constitutes up to 80% of overall email volume [1]. This results in a waste of bandwidth and productivity for companies, and a waste of time and an annoyance for email users.

As the problem has grown, a number of methods have been developed to detect and stop spam before it reaches the inbox. The main techniques employed are:

- IP Address blacklisting,
- Phrase Matching,
- Statistics, and
- Heuristics

IP Address blacklisting [1, 2] works by storing a list of the origin of known spam and then ignoring further email sent from that IP address, under the assumption it is also spam. This approach has two problems: spammers are able to circumvent it by regularly switching IP addresses, and after spam has been sent from an IP address hijacked by a spammer, email sent by the unsuspecting computer user is now blacklisted as spam as well.

Phrase matching [1] examines incoming email, looking for phrases that have been used in mail that has already been identified as spam. This works reasonably well at identifying spam that has already been circulated, but falls down when trying to identify new spam.

Statistical approaches [1, 3] work by learning what a particular user’s personal email looks like. They extract features from the email and determine how many times each feature has been recorded in normal email and spam. The probabilities for each feature are then combined into a final probability that determines whether the email is detected as spam or not. This works particularly well for personal email, but the features of the user’s email must first be learnt before it becomes effective.

The Heuristics method [3] tests an email against a large number of rules, each designed to test whether an email is spam or not. Each rule is generally given a weighting, and then a score for the email is generated by summing the weightings for all the tests that it failed. If the email’s score is over a certain threshold, it is then declared to be spam. This method, along with the statistical method, have proven to be the most effective in detecting spam. The table below, based on data from [3], gives a rough idea of the effectiveness of each of the methods.
<table>
<thead>
<tr>
<th></th>
<th>IP Address Blacklisting</th>
<th>Phrase Matching</th>
<th>Statistics</th>
<th>Heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0-60%</td>
<td>80%</td>
<td>80-99%</td>
<td>95-99%</td>
</tr>
<tr>
<td>False Positives</td>
<td>10%</td>
<td>2%</td>
<td>0.1%</td>
<td>0.5%</td>
</tr>
</tbody>
</table>

Whilst heuristics has proven to be the most effective method overall, the problem with this method is determining what weighting to give each rule in order to maximise the accuracy in identifying spam. Due to the large number of rules employed, mathematical techniques are too unwieldy and an exhaustive search is too time-consuming. Therefore, other optimisation techniques need to be used. One technique that could prove useful is Genetic Algorithms.

Genetic algorithms (GAs) are a computer-based method of optimisation that use a process imitating natural evolution in order to solve a problem. The basic process is as follows: First, a number of possible solutions (a ‘population’) are generated randomly. These solutions are then tested, and the best solutions are selected and the others discarded. The selected solutions are then recombined in some way to create a population of ‘child’ solutions, that combine characteristics from one or more of the ‘parent’ solutions and are also randomly altered, or ‘mutated’ slightly, and thus while similar to their parents, they are not identical. These ‘child’ solutions are then tested, and again the best solutions are selected for further ‘breeding’. This process continues until an acceptable solution to the problem has been reached.

GAs are most useful in optimisation type problems for which there is no obvious way to use more traditional problem solving techniques [4]. All that is required to tackle the problem using a GA is a way of representing a solution to the problem, and a way of testing and ranking potential solutions. Given this information, a GA can be constructed to create and evolve a set of solutions until a good solution to the problem is reached.

This characteristic makes GAs useful for finding the optimum rule weightings for heuristic spam detection: a genome of a possible solution is constructed as a list of weightings for each rule, and a possible solution can be tested by running the detection engine (using the weightings specified by the solution) on a large number of ‘training’ emails for which the results (i.e. whether it is spam or not) are already known.

Indeed, a GA has been used successfully in email detection by the program SpamAssassin. SpamAssassin is an email filter that uses a heuristics based spam detection engine to identify spam. Version 2.6 uses 872 rules to test spam and the latest version 3.0 uses 628 [5]. SpamAssassin is one of the leading email detection programs currently available, and has received numerous awards [3]. A previous version of SpamAssassin (version 2.6) used a GA to calculate the optimum weightings. This was replaced in version 3.0 by a Perceptron Learner, which is a type of neural network. The main reason given for this change was that the GA, which was developed a number of years previously, had not been maintained and the current development team did not understand how it worked [5]. Indeed, the Perceptron Learner only performed on an equal footing to the GA, but did prove to be faster.

Although GAs do perform quite effectively in this case, as demonstrated by SpamAssassin, performance could perhaps be improved by using a Memetic Algorithm (MA). A Memetic Algorithm combines the principle of Genetic Algorithms with a local search, which is used to perform local optimisation – something that GAs struggle with. Whilst GAs can quickly find a good solution to an optimisation problem, they are not good at finding the perfect solution.
MAs can use a local search to quickly find if there are any solutions ‘close’ to the solutions in the population that provide a better result. This can help the algorithm to move from a good solution to the optimum solution. MAs have been shown to be more effective than GAs for a number of problems [7].

A further complication of spam detection is that the problem has not one, but two objectives which must be considered in the optimisation process. Whilst a spam filter aims to reduce the amount of spam escaping detection, it must also ensure that the number of legitimate emails that are marked as spam are minimised. These two objectives conflict, as increasing the strictness of the filter to stop more spam will likely also lead to more legitimate email being marked as spam. Similarly, relaxing the filter to ensure all legitimate email gets through will also lead to more spam escaping detection. Therefore, spam detection is in fact a multi-objective problem (MOP), as it has two conflicting objectives, and improving the performance for one objective requires a trade-off in the performance of the other.

It is in fact possible to reduce any MOP to a single objective, which can then be optimised using single objective optimisation techniques [8]. This can be done by combining all the objectives into a single function, weighting each based on its perceived importance. The resulting function can then be treated like a single objective, and optimised in that way. However this approach is of limited usefulness for spam detection, as the optimised solution found only applies to that particular case, and in order to find the optimum solution in another case, the evaluation function needs to be adjusted and then the optimisation process run again.

The importance of the two objectives of spam detection can vary widely between users: for many users it is far more critical to ensure that all legitimate email are received, and they are willing to accept some spam, rather than risk missing an important email. An example of this would be a business user, who would like to ensure they receive all emails sent by their customers. For other users it is important to avoid receiving any spam, even at the risk of missing some legitimate emails. An example of this case is a child’s email account – which should be protected from the offensive content found in many spam emails. In order to optimise spam detection under a variety of situations a method of finding the set of optimal solutions is needed.

In order to find the set of optimal solutions for a MOP, the notion of Pareto optimality is usually adopted [8]. A Pareto optimum is a solution to a MOP for which no other feasible solution exists that improves performance for one objective without simultaneously decreasing it for another objective. The set of all such optimal solutions forms the Pareto front. The benefit of using GAs or MAs to solve MOPs is that, unlike most optimisation techniques, only a single run of the algorithm is needed to find the front of Pareto optimal solutions. Simplistically, this is achieved by selecting from the population all those solutions which are Pareto optimal with respect to the rest of the population. This gives a set of solutions that, over a number of generations, should approach the Pareto front, and thus optimise the MOP. This approach would allow the performance of a spam filter to be optimised for all situations simultaneously, and then the optimal solution selected depending on user preference for spam prevention or reduced false positives.
**Aim**

There are two main aims of this project:

- To investigate whether the use of a Memetic Algorithm can improve on the Spam detection performance of Genetic Algorithms.
- To utilise and study the multi-objective approach in construction of the algorithm, so that spam detection is optimised over all conditions.

If time permits, an area of further interest would be to extend the algorithm to allow for online learning – that is to continue to improve the spam filter based on the user’s actual email.

**Method**

In order to achieve the aims of this project both a multi-objective GA and a multi-objective MA will need to be developed, trained and tested. The algorithms will be designed to alter the weightings of rules used in heuristic spam detection in order to optimise the performance of spam filtering.

The SpamAssassin anti-spam software will be used as the framework with which to train and test the algorithms. SpamAssassin will handle the actual spam detection process, including message parsing and testing of emails using its inbuilt message parser and heuristic spam detection engine. It will be necessary to develop an interface to SpamAssassin to allow the weightings of rules from its spam detection to be altered, and to provide a means of training and testing the algorithms. A large corpus of email, containing both known legitimate email and known spam, must also be sourced. This will be required to provide data so that the algorithms can be both trained and tested.

A basic schedule of the required activities and the expected timeframe to complete each is given in the table below.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learn about, and code a simple Genetic Algorithm</td>
<td>March</td>
</tr>
<tr>
<td>Learn about and code a Multi-objective Genetic Algorithm</td>
<td>March – April</td>
</tr>
<tr>
<td>Find a source of emails to use as training and testing data</td>
<td>April</td>
</tr>
<tr>
<td>Develop means of altering weights of SpamAssassin rules, and to run SpamAssassin on training and test emails</td>
<td>April</td>
</tr>
<tr>
<td>Develop Genetic and Memetic Algorithms to adjust weightings of rules</td>
<td>May – June</td>
</tr>
<tr>
<td>Run experiments to determine performance of algorithms (against each other and the SpamAssassin defaults)</td>
<td>July</td>
</tr>
<tr>
<td>Write draft dissertation</td>
<td>August – September</td>
</tr>
<tr>
<td>Finalise experiments</td>
<td>September – October</td>
</tr>
<tr>
<td>Complete dissertation and prepare presentation and poster</td>
<td>October</td>
</tr>
</tbody>
</table>
References


