Assessing the interestingness of emails with Ant Colony Algorithms

Allen Chan
University of Kent
Canterbury
Kent
ac28@kent.ac.uk

Abstract

With the success of the Internet, the use of electronic mail (e-mail) has increased dramatically as a source of communication and with access to higher bandwidth connections, e-mails has become an integral part of our lifestyle. With every e-mail we receive, there is an element of interestingness associated with e-mails to its user. This paper looks at adapting an ant colony based data mining algorithm for classifying e-mails with respect to their degree of interestingness. More precisely, each e-mail is assigned one of the three following classes: high, medium, or low degree of interestingness.

1. Introduction

Since the global introduction of the World Wide Web in the late 1980’s, its popularity has experienced exponential growth and so has many of the applications that the Internet is providing. The Internet has increasingly provided users with many applications, and one of the major applications available via the Internet is sending electronic mail (e-mail). With the ever increasing amount of e-mails that users receive, a common routine but tedious task that is carried out by all users is to categorise incoming e-mails from the inbox into subfolders. Where each subfolder represents a separate area of interest, e.g. sport, business etc.

The task of manual classification by user’s is already becoming a long and time consuming task, and with the exploitation of the Internet by companies and certain individuals, mixed in conjunction with genuine e-mails is what is known as spam mail (unsolicited bulk mail), which is uninteresting and irrelevant to the user. There is already sophisticated software available on the software market for dealing with spam mail. As the classification rules for spam mail is different to that of normal e-mail classification, this research focuses on the classification of genuine e-mails users receive in their inbox.

The traditional task of classification is by using a data mining algorithm trained on an existing set of examples to discover useful rules to classify unseen examples. E-mail classification is a specialised form of data mining called web mining which only deals with content that exists on the World Wide Web. The task of e-mail classification is to sort e-mails into user specified folders. This research differs from traditional e-mail classification with respect to two points. First, the research looks at an ant colony optimization algorithm (ACO) to classify e-mails. Second, rather than looking at the traditional view of classifying e-mails into set user specified folders; the e-mails will be classified according to how interesting they appear to the user. This will give us an insight into whether the ACO algorithm will be able to deal with a users change in interests over time.

The algorithm is based on an ant colony optimization algorithm for data mining called Ant-Miner as detailed in [1]. This research alters the Ant-Miner algorithm to deal specifically with e-mail classification for this particular area of research.

The rest of this research report explains in section 2 the task of e-mail classification and challenges that exist in this area of research, and also previous classifiers that has been developed. Section 3 explains the ACO and Ant-Miner algorithms. Section 4 will report on the experimental setup and results that the algorithm has achieved. Section 5 will open discussions to barriers encountered during this research and conclusions this research work exhibits.

2. E-mail classification

E-mail classification is a sub domain of a research area in computer science known as data
mining (also known as knowledge discovery in databases, KDD). The aim of KDD is a process to discover ‘useful’ information from the abundance of data that exists on the Internet and data held in databases of large organisations and companies. For further detailed information on what KDD entails, the reader is suggested to refer to [2].

This research work is aimed at a more specific kind of information that exists globally, namely the World Wide Web. Web mining as it is commonly referred to as, is specifically targeted to discover useful knowledge from information that can be obtained through the Internet, which includes text, documents, images, audio and e-mails, which is our primary source of data in this research.

The main focal point of this research work is on the classification of e-mails. Current research on classifying e-mails is further explained in the following sections.

2.1. The classification task: a brief

In general, the classification task is for a selected KDD algorithm to discover information in the form of rules that is useful for classifying new (or unseen) examples that the algorithm has not seen.

The classification task consists of predicting the class of an example (record) based on the values of set of predictor attributes describing that example. In the classification task, the data being mined is divided into two data (sub)sets, namely the training set and the test set. The classification algorithm has to discover a classification model (to predict the class of examples) based on the training set only. Once the classification model has been discovered, it is applied to classify examples in the test set, which were not seen by the classification algorithm during its training. The goal is to obtain a high classification accuracy in the test set, where classification accuracy is defined as the number of correctly classified test examples divided by the total number of test examples.

The classification task can be described by the following 3 stages:

- Pre-process data (e-mails, in this work) into a suitable dataset containing predictor attributes and a class to be predicted.
- Pass the pre-processed dataset to a classification algorithm for classification rules to be discovered.
- Apply discovered classification rules to classify a set of unseen examples.

In the first stage the dataset is generally produced in a table like format, where the first row contains, for each column, an attribute name, and each of the subsequent rows representing a single record in the dataset (a single e-mail, in this work) with the corresponding values for each attribute. The last column is used to hold the class which an example fits into.

Once the pre-processed dataset is ready, the classification algorithm will take the dataset and start to discover one or more classification rules which will be used to classify unseen examples in the third stage. Each rule consists of a rule antecedent (a conjunction of terms) and a consequent which is the class predicted by the rule. Each rule discovered is in the following form:

\[
IF \ <term1> \ AND...AND \ <term-m> \ THEN \ class
\]

where each term is of the form \(<attrib=value>\). Each rule will vary, and a rule can contain one or more terms, but no contradicting terms e.g. \(<sex=m> \ AND \ <sex=f>\).

The third and final stage is the actual classification task. This is to actually apply the discovered rules to an unseen set of examples and/or real examples.

2.2. The challenge of e-mail classification

Traditionally, e-mail classification has been a manual task, and with the increased growth of popularity of e-mails, there is also an increased amount to classify. This is one of the reasons why increased interest in an automatic e-mail classifier is preferred. Current research looks into the classification of e-mails into user specified folders, although this seems to be theoretically simple, there are certain challenges that needs to be addressed when implementing a suitable algorithm to deal with the problem domain of e-mail classification.

The main challenges that occur in e-mail classification which are related to this research are outlined as follows. Further detailed challenges can be found in a previous discussion by J.D. Brutlag and C. Meek [3]:

- A large number of attributes, words in the case of e-mails.
- Dynamic concepts of e-mails.
For the classification process, within an e-mail, attributes must be selected to be used by the algorithm for training and classification. In previous research there has been a suggested combination of words to be extracted and then transformed to a list of attributes. Every e-mail consists of a sender, subject and body:

- **Sender** – contains information of the sender i.e. e-mail address, a nickname that the sender uses to be displayed in receiver mail client, and some e-mails sent may contain the date and time that the e-mail was sent.
- **Subject** – contains either key words or a short succinct sentence about the topic of the e-mail.
- **Body** – contains the main message sent by the sender.

If all 3 sections of an e-mail were to be considered by the classification algorithm, then a collection of e-mails would lead to an increasingly large number of attributes to be processed. In previous research, the sender and subject fields have been shown to yield similar results to using the main body of an e-mail [4]. Therefore to overcome this problem of an increasing large list of attributes, this work only deals with the sender and subject of an e-mail.

As time changes, a user’s classification habits may also change with time. The main concern here is the change in interest where a user may change his/her hobbies, current dealings, and change in priorities of daily tasks. For the chosen algorithm that this work deals with, the possibility of the discovered rules to deal with the dynamic aspect of e-mails depends on how vast the user’s interests extend and how frequent the change occurs. If a user’s interest changes too frequently then the chances of the algorithm being able to cope with the change is small. However, with the expectation that a user’s interests change occasionally, then the algorithm should be able to cope with the changes.

### 2.3. Related work on e-mail classifiers

Automating e-mail classification is not a particularly new area of research, there has already been research into e-mail classifiers. Each of these work focuses on a specific algorithm for classifying e-mails and each have shown interesting results.

One of the classifiers called ‘The Intelligent E-mail Sorter’ (IEMS) [5], uses an instance based learner combined with a general explicit description for classification referred to as a composite rule learner which achieves an accuracy of between 55%-65%.

Another e-mail classifier the MailCat [6], uses a TF-IDF approach text classifier to classify e-mails. The actual program not only classifies e-mails into one class, but offers the user several choices to choose from. The accuracy of this algorithm claims to be between 80%-90%, but by offering three classes rather than one. But this e-mail classifier only automates e-mail classification to a certain extent as the user still has to provide the final folder that the e-mail should go into.

The Artificial Immune System for E-mail Classification (AISEC) [7], is an algorithm inspired by the artificial immune system, and uses a completely different approach to conventional algorithms for data mining. The accuracy achieved by the AISEC is approximately 90%. But this algorithm not only classifies e-mails, but also implements a continuous learning memory. But when compared to a well known classifier, the Bayesian algorithm (which was retrained periodically to cope with the dynamic aspects of the e-mails) proved to match the accuracy of the AISEC.

### 3. ACO and Ant-Miner

An ant colony optimization algorithm is based on the natural behaviour of a swarm of ants (known as swarm intelligence). The idea of an ACO algorithm uses the observed shortest route finding by a swarm of ants from a nest to a source of food, as explained in [8].

The Ant-Miner algorithm as detailed in [1], is an adaptation of the ACO algorithm especially for data mining. The algorithm implements the aspects of awarding attributes used most by the ants with pheromone, which increases the probability of that attribute being selected by the next ant. Every ant creates a rule and, within a population of ants, the best rule is then selected as a discovered rule sorted to be used for classifying unseen examples. Ant-Miner is described in more detail in Section 3.2.

### 3.1. An overview of ACO

The ACO algorithm was first proposed by Marco Dorigo and his colleges to solve difficult
combinatorial optimization problems like the classic travelling salesman problem [9]. The algorithm was developed after observing the unique foraging behaviour to finding the shortest path between two points in real ant colonies. In real ant colonies, as each ant is walking between its nests to a food source, it deposits a chemical trail of pheromone which is recognised by itself and other ants within the colony. The pheromone in the trails followed by each ant and its nest mates increases and trails with a higher concentration of pheromone have an increased probability that other ants will follow. It has been proven in an experiment set up by Deneubourg at al [10], that a shortest path followed by a colony of ants emerges over time. This behaviour has led to an increased interest in developing the ACO algorithm for solving problems like the TSP.

The key ideas that an ACO algorithm inherits from real ant colonies is the indirect communication by pheromone laying also known as stigmergy (where communication is made by altering the physical environment in such a way that only the insects themselves will recognise the alteration). Another key idea is the positive feedback mechanism with implicit solution evaluation where the shortest path between two points is found first, and over time the pheromone along that path will increase faster than that of a longer path. As the pheromone in the shorter path is increased faster, the colony converges on that path and hence a solution for the shortest path is found. This is achieved by the positive feedback through the increased pheromone depositing. Further detailed description of the behaviour can be found in [9].

In an ACO algorithm, a finite size colony of ants is used, and each ant finds a solution or part of a solution according to some criteria to the given problem domain. Each ant uses the information that it has collected to calculate a performance indicator to update global states (represented by a pheromone matrix) initially defined before the first ant starts to forage for a solution. This updating modifies how the problem is represented as seen by other ants within the colony. As one of the key ideas that is mentioned, ants do not communicate directly, they do so by stigmergy (indirect communication), and the main feature of ant colonies is the updating of pheromone along the path a particular ant has chosen to follow.

Each solution found by an ant must follow the constraints set out for the problem at hand. Each ant is capable of finding a poor solution; it is after a short period of time before a good solution emerges due to the global exchange of information detected by each ant.

The solutions are created by the ants by moving through a sequence of neighbouring states, which are selected based on a combination of a local search policy (involving a priori information about the problem domain) and the globally available information related to the amount of pheromone associated with the search states.

After a solution is found by an ant, the ant will update the globally available information, namely the pheromone trails, according the quality of the solution it has found. Once the updating has finished, the ant is destroyed as it no longer has any use within the future operation of the ACO algorithm. As explained before, it is over time, depending on how big the finite search space is in the problem domain, that one or more good quality solutions are found.

To summarise, an ACO operates three stages, the first being the generation of a finite size colony of ants. The second stage is setting the ants to search for a solution, where the ants are guided by pheromone and any other problem related information that will affect the next ants’ decision. The final stage is the updating and evaporation of pheromone after analyzing each solution generated by each ant.

3.2. Ant-Miner

The main aspects of an ACO algorithm are already explained in the previous section. The Ant-Miner is an adaptation of the ACO algorithm especially for data mining [1]. Pseudocode.1 illustrates the Ant-Miner algorithm used in this research.

In general, in an ACO algorithm, each ant constructs and/or modifies a solution for the target problem. In the case of the Ant-Miner, the target problem is to discover classification rules, and as mentioned earlier in section 2.1 every rule must be in the form of:

\[
IF <term1> AND...AND <term-m> THEN class
\]

Each term takes on the form of \(<attribute, operator, value>\), where value belongs to the domain of the attributes and the operator is a relational operator connecting both attribute and value. The version of Ant-Miner this research uses can cope with only categorical attributes,
The only operator within each term discovered will always be ‘=’.

TrainingSet = \{all training cases\};

DiscoveredRuleList = \{\};

WHILE (TrainingSet>MaxUncoveredCases)
  a = 1; //ant index
  conv = 1; //convergence test index
  Initializes all trails with equal amount of pheromone;
  REPEAT
    Ant_\textsubscript{a} starts with an empty rule and adds one term at a time to the current classification rule being constructed till R_\textsubscript{a} covers less than a certain number of examples;
    Prune newly constructed rule R_\textsubscript{a};
    Update pheromone trails increasing pheromone in the path followed by current ant Ant_\textsubscript{a}, and decreasing pheromone in those paths not followed;
    IF(R_\textsubscript{a} is equal to R_\textsubscript{a-1}) //convergence test
      THEN conv = conv + 1;
      ELSE conv = 1;
    END IF
    a = a + 1
  UNTIL(a\geq NoOfAnts)
  OR (conv\geq NoRulesConv)
  Choose best rule R_{best} from all rules constructed by all ants;
  Add R_{best} to DiscoveredRuleList;
  TrainingSet = TrainingSet-{examples covered by R_{best}};
END WHILE

Psuedocode.1

The Ant-Miner algorithm uses a sequential covering approach to discover a list of classification rules which will cover all or most of all the examples in the training set. Before the algorithm is applied to the training set, there is a pre-processing step applied to calculate the probabilities of each term using a problem heuristic function which is taken from [1]. When the algorithm first starts, the training set holds all the training examples and the discovered rule list is empty. Every iteration of the WHILE loop illustrated in Psuedocode.1, creates a population of ants, each ant corresponding to one iteration of the REPEAT-UNTIL loop, also where each ant constructs one rule. At the end of the WHILE loop, the best rule from the set of constructed rules is added to the discovered rule list. Examples covered by this rule are removed from the training set before the next WHILE loop begins to discover the next rule. This rule discovery process is repeated until the number of uncovered examples in the training set is less than a user specified threshold (MaxUncoveredCases).

Every iteration of the REPEAT-UNTIL loop consists of three stages; rule construction, rule pruning, and pheromone updating, explained as follows. Every Ant_\textsubscript{a} starts off with an empty rule with no term in its antecedent, and adds one term at until one of two criteria is met:

- Any term added to the current rule R_\textsubscript{a} would make the rule cover a number of examples less than a user specified threshold (MinCasesPerRule).
- All attributes have been used by the current ant Ant_\textsubscript{a}, which means there are no more terms which can be added to the rule antecedent. As mentioned earlier, no rule can contain any attribute twice, e.g. \textless sex=m \textgreater AND \textless sex=f \textgreater .

The current partial rule being constructed by Ant_\textsubscript{a}, represents the path being taken by that ant, and every term added to the current partial rule constitutes the direction of how the path is being extend. Every term that is chosen is selected using a roulette wheel mechanism, where a term with a higher probability will have a higher chance of being selected. The probability of a term being selected depends on both a problem dependent heuristic function and the amount of pheromone associated with it.

After the rule construction stage, every rule R_\textsubscript{a} then undergoes rule pruning, where the aim is to remove all irrelevant terms and also to improve the predictive power of the current rule R_\textsubscript{a}. This process is necessary as some of the terms added to the rule antecedent may have been added due to stochastic variations in the
Pheromone increasing and decreasing are explained respectively as follows.

As mentioned already, only those terms which occur in the rule antecedent have their pheromone updated, to simulate pheromone deposition by the current ant. The amount of pheromone added to a path followed by the current ant is given by Equation.3. Once all paths followed have had their pheromone levels increased, the paths not followed will then (indirectly) have their relative pheromone levels decreased.

\[ \tau_i(a + 1) = \tau_i(a) + \tau_i(a) \times Q \]

**Equation.3**

More precisely, after pheromone updating all the pheromone values are normalized, by dividing each \( \tau_{ij} \) by the summation of all \( \tau_{ij} \). Hence, terms that did not have their \( \tau_{ij} \) increased during pheromone updating will effectively have a reduced \( \tau_{ij} \) value after normalisation.

After all three stages have been completed; this is the end of the REPEAT-UNTIL loop. This loop is repeated till one of the following terminating criteria is met:

- The number of constructed rules is equal or greater than the number of ants specified by the user.
- The rule constructed by Ant \( a \) is exactly the same as the rule constructed by the previous NoAntConv (number of ants for convergence, a user-specified parameter) ants. This criterion is checked via a convergence test.

4. Results

To test the classification accuracy of the rules discovered by the Ant-Miner algorithm, it was necessary to compare the results with another rule discovery algorithm, to see how well the Ant-Miner algorithm performed. The chosen algorithm is C5.0, which is a well-known rule induction algorithm used widely in industry.

As with traditional classification, the aim of discovered rules is to correctly classify unseen examples into a class predicted by the discovered rules. Each e-mail in the dataset was manually classified as one of three classes: high, medium, or low degree of interestingness. Due
to the difficulty of finding a public-domain set of e-mails pre-classified into the three different interestingness classes, the author has manually classified his own e-mails for the experiments conducted in this research.

4.1. Experimental Setup

Experiments were performed with 500 e-mails in total. Of the 500, 174 (34.8%) were classified with a 'high' degree of interestingness, 198 (39.6%) were classified with a 'med' (medium) degree of interestingness, and 128 (25.6%) were classified with a 'low' degree of interestingness. The dates of the e-mails ranged from January 2002 all through to December 2003, and the temporal ordering was preserved, otherwise there would not have been an evident change of interest over time. Only the words from the sender and subject sections of the e-mail were used and each of these was tokenized removing any characters other than the letters of the alphabet to leave only words that occurred in the e-mails. All these words collected together constituted all the attributes of the dataset, i.e. each attribute represents the presence or the absence of the corresponding word in an e-mail.

In conjunction to removing numbers and punctuations from the sender and subject lines of each e-mail, a list of stopwords was used to reduce the number of attributes that had been produced from the previous step. During pre-processing of the dataset, after extracting all words from both the sender and subject line, there were an increasingly large number of words that would become attributes – even after removing stopwords. Therefore a further step in the pre-processing of the dataset was needed. Hence, attributes were created only for words that occurred in all the five partitions of the data. This avoids that, in a given partition, an algorithm have the constant value "word = no", which would render that attribute irrelevant for classification purposes.

The algorithm as described in section 3.2 requires the specification of several user-specified parameters. The parameters and their values chosen for this experiment are listed in Table.1. These values were set to the default values used in [1].

Two experiments were conducted, each using a different method. The first experiment used 5-fold cross-validation. That is, the dataset was split into 5 partitions (each with 100 examples), and both algorithms were trained using 4 of the 5 partitions and the discovered rules were used to classify examples in the partition that was not used to train either algorithms. This process was repeated five times. Table.2 shows the assigned names of the training set and test set for easy reference in the rest of this paper.

4.2 Classification Results

After training both algorithms, the collected results of the first experiment are shown in Table.3. These results report the classification accuracy on each partition (set A-E) that was not in the group trained by the algorithm.
Apart from reporting the classification accuracy, the number of rules discovered and also the number of terms per rule is also measured for each of the groups trained by both algorithms. These results are illustrated in Table 4.

Overall Ant-Miner performs better than C5.0 with respect to both predictive accuracy and rule set simplicity, as can be observed in the last rows (average results) of Table 3 and Table 4. In particular, in the last row of Table 4 the total number of terms discovered by Ant-Miner was 39.6 (9 * 4.4), whereas the total number of terms discovered by C5.0 was 47.36 (12.8 * 3.7).

<table>
<thead>
<tr>
<th>Group</th>
<th>Rules</th>
<th>Terms/Rule</th>
<th>Rules</th>
<th>Terms/Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>7.3</td>
<td>12</td>
<td>2.5</td>
</tr>
<tr>
<td>B</td>
<td>8</td>
<td>4.1</td>
<td>13</td>
<td>4.9</td>
</tr>
<tr>
<td>C</td>
<td>9</td>
<td>2.2</td>
<td>13</td>
<td>3.7</td>
</tr>
<tr>
<td>D</td>
<td>9</td>
<td>2.2</td>
<td>14</td>
<td>3.2</td>
</tr>
<tr>
<td>E</td>
<td>9</td>
<td>4.4</td>
<td>12</td>
<td>3.5</td>
</tr>
<tr>
<td>Ave</td>
<td>9</td>
<td>4.4</td>
<td>12.8</td>
<td>3.7</td>
</tr>
</tbody>
</table>

Table 4

The result of the second experiment is reported in Table 5 and Table 6. As can be observed in Table 5, the C5.0 algorithm performs slightly better in most cases when compared to the Ant-Miner algorithm. Apart from the last partition, the difference between C5.0’s accuracy and Ant-Miner’s accuracy is less than 1%.

<table>
<thead>
<tr>
<th>Train/Test partitions</th>
<th>Ant-Miner</th>
<th>C5.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 / 2</td>
<td>37%</td>
<td>37%</td>
</tr>
<tr>
<td>2 / 3</td>
<td>39.4%</td>
<td>40%</td>
</tr>
<tr>
<td>3 / 4</td>
<td>49.2%</td>
<td>50%</td>
</tr>
<tr>
<td>4 / 5</td>
<td>41.6%</td>
<td>53%</td>
</tr>
<tr>
<td>Ave</td>
<td>41.8%</td>
<td>45%</td>
</tr>
</tbody>
</table>

Table 5

In the second experiment, as can be observed in Table 6, Ant-Miner produces a smaller set of rules and also with a smaller number of terms per rule. This supports previous experiments conducted using the Ant-Miner algorithm for data mining [1] where rules discovered are more comprehensible than those produced by other algorithms used for classification.

<table>
<thead>
<tr>
<th>No. of rules and terms discovered</th>
<th>Ant-Miner</th>
<th>C5.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitions</td>
<td>Rules</td>
<td>Terms/Rule</td>
</tr>
<tr>
<td>1 on 2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>2 on 3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3 on 4</td>
<td>4</td>
<td>1.2</td>
</tr>
<tr>
<td>4 on 5</td>
<td>4</td>
<td>1.2</td>
</tr>
<tr>
<td>Ave</td>
<td>4.25</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Table 6

5. Conclusion

The aim in this research was to investigate the performance of an ACO algorithm in web mining. In particular, the task was e-mail classification with respect to the degree of interestingness of e-mails. In terms of classification accuracy, it is shown here that Ant-Miner is competitive with the popular industry standard C5.0 algorithm with respect to predictive accuracy. Also, an interesting finding was that the rule set discovered by Ant-Miner was smaller than the rule set discovered by the C5.0 algorithm. Hence, Ant-Miner offers close predictive accuracy with a smaller discovered rule set, which can be more succinct for human interpretation of discovered knowledge.

Within the discovered rules by both algorithms, the first of the Ant-Miner rules started off with a large number of terms in its rule antecedent, and later rules were reduced to only one or two terms. With the C5.0 algorithm, the rules contained at most three to four terms in the rule antecedent.

Concerning the time it takes to discover the rules. C5.0 is by far a much quicker algorithm. Ant-Miner takes a much longer time, which is mainly due to its computationally-expensive rule pruning process.

From the experiments conducted, as both algorithms have shown a low predictive accuracy, this indicates that it is not the Ant-Miner algorithm which is at fault. But suggests there are other factors involved (like data pre-processing techniques used and choice of words to be used as attributes) which is preventing both algorithms from achieving a higher accuracy.

5.1. Barriers

During this research, a major barrier that was encountered was the large number of attributes in the data set. One reason for such a large set of attributes to occur is down to the vast variations in a user’s choice of words to summarise an e-
mail to be sent. There are e-mails that users will not include any description in the subject line, and where there is some form of description, spelling mistakes must be accounted for and also the large range of abbreviation of words that users use. This has shown the simple data reduction techniques are not sufficient during the data pre-processing stage. In addition to the simple data reduction techniques used in this paper (such as stopwords removal), more powerful data reduction techniques could be used in the future, to improve the results.

This shows that it is important to pre-process the data into a suitable representation for the classification algorithm, in order to achieve a high predictive accuracy. All the attributes and values selected should best describe the data within the dataset.

5.2. Further Work

Following this research, there are some further extensions which are still open for research. First, rather than using just the sender and subject lines of an e-mail, the main body could be included for classification to see if the accuracy improves. Although a major problem will be the extremely large set of words which can be used as attributes. This leads to the second area of further research. As already shown through this research, with e-mail classification, simple data reduction techniques have not shown to be very effective in producing a high predictive accuracy. Therefore an area where further work can be carried out is to look into more powerful data reduction techniques for the classification task in general, and also more specifically for web-content mining.

Thirdly, it would be interesting to investigate if it is possible to set up multiple ants to discover rules in parallel (like in a real ant colony) increasing the speed of the Ant-Miner algorithm. Currently one ant at a time is used to create a rule and then being destroyed, which is a time consuming task.

6. References


