24

A Knowledge Acquisition Method of Judgment Rules for Spam E-mail by using Self Organizing Map and Automatically Defined Groups by Genetic Programming

Takumi ICHIMURA, Kazuya MERA and Akira HARA
Hiroshima City University
Japan

1. Introduction

Recently, the Internet has been a basis of our cultural life. Web provides a virtual huge space of information, where an emotional experience, a political idea, a cultural custom, and the advice of manners of music, the business, the arts, photographs, and literatures, etc. are digitalized at a low price and are shared for our current culture. One of major other tools gives E-mail communication which propagates not only letters from person to person, but a means of advertisement. However, E-mail addresses on the web page are gained and the virus is appended to E-mails, and the attacks for acquiring information on user's personal computer (called BOT) have been spreaded. Their accumulated E-mail addresses collected in such a way will be valuable for advertising agents. But, their E-mails are called “Spamming“ and the Spamming becomes a one of the social issues.
Spamming is the abuse of electronic messaging systems to send unsolicited bulk messages or to promote products or services, which are almost universally undesired. Spamming is economically viable because advertisers have no operating costs beyond the management of their mailing lists. The sender cannot be specified, because the sender of Spamming has only temporary E-mail address and the reply of them is not reached to the original sender. Therefore, undesired E-mails to us have been increased everyday, so that, it is not easy to read an important E-mail.
In order to avoid Spam E-mails, we must build the filtering system which can judge whether the received E-mail is a Spam E-mail or not. The SpamAssassin(SpamAssassin) is the open source software and is a mail filter which attempts to identify a Spam E-mail using various pattern match methods including text analysis, Bayesian filtering, DNS block lists, and collaborative filtering databases. There are the predefined rules for each filtering method to detect a Spam E-mail. The agreement degree for each rule is scored and if the total score is larger than the threshold value, the given E-mail is judged as Spam. Although each score in the rule of SpamAssassin is low, there is a few cases in which total score becomes high. Moreover, even if the message is judged as Spam, the different rules for each person are

required according to the personal environment such as work style, because a content of received message will be different.
In this paper, we propose a classification method for Spam E-mail based on the results of SpamAssassin, which is the open source software to identify spam signatures. This method can learn patterns of Spam E-mails and Ham ones and correctly recognizes them. First, the method divides E-mails into some categories by Self-Organizing Map (SOM) (Kohonen, 1995) and extracts the adequate judgment rules by Automatically Defined Groups(ADG) (Hara, 1999), even if the judgment results by SpamAssassin are wrong.
The SOM is developed by Kohonen and is a topology-preserving map because there is a topological structure imposed on the nodes in the network. A topological map is a simple mapping that preserves neighborhood relations. The SOM is an algorithm used to visualize and interpret large high-dimensional data sets.
The ADG is a new method that united Genetic Programming (GP) with cooperative problem solving by multiple agents. By using this method, we had developed the rule extraction system from database (Hara, 2004, 2005, 2008). In this system, two or more rules hidden in the database and respective rules' importance can be acquired by cooperation of agents.
In order to verify the effectiveness of our proposed method, about 3,000 Spam and Ham E-mails are examined. The SpamAssassin works in the Linux Server where Postfix(Frederick) operates as Mail Transfer Agent (MTA) and the interface between MTA and content checkers is amavisd-new (Amavisd-new). Section 2 describes the operation of SpamAssassin and Postfix with an interface called amavisd-new. Section 3 and 4 explain the mechanism about SOM and ADG. Section 5 described the experimental results. Section 6 gives the conclusive discussion.

2. SpamAssassin under MTA

The SpamAssassin is a flexible and powerful set of Perl programs which score the agreement degree from multiple types of checks to determine whether a given E-mail is Spam. Because the SpamAssassin has no function to receive or send an E-mail, it must operate with MTA such as Postfix. Fig.1 shows the operation between Postfix and SpamAssassin through Amavisd-new. The clamav in Fig.1 is the Clam AntiVirus (Clam AntiVirus) which is a GPL anti-virus toolkit for UNIX, designed especially for E-mail scanning on mail gateways.
The SpamAssassin has the following features:
- Header tests
- Body phrase tests
- Bayesian filtering
- Check of E-mail address of blacklist/whitelist automatically
- Check of E-mail address of blacklist/whitelist manually
- Tests by using Collaborative Spam identification databases
- Tests by using DNS Blocklists
- Tests of Character sets and locales

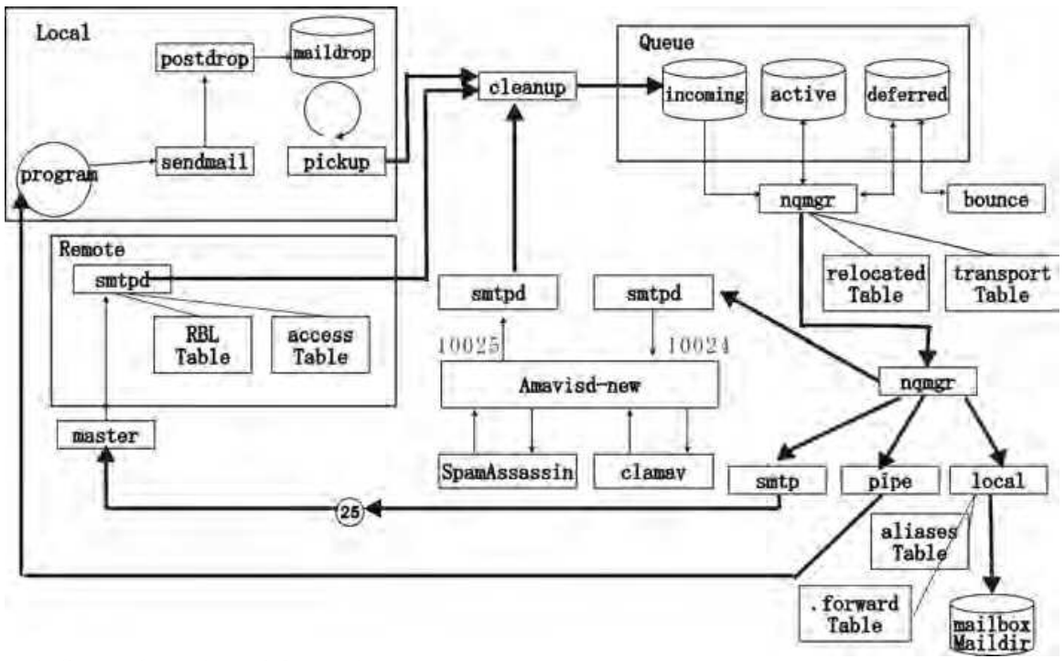

Fig.1 A flow of MTA and SpamAssassin

Even if any one of these tests may not identify a Spam or a Ham correctly, it is not easy to make the correct judgment by only their combined score. For example, Fig.2, Fig.3, and Fig.4 show the headers of E-mails which are judged as Spams through the SpamAssassin.

```
X-Spam-Flag: YES
X-Spam-Score: 57.834
X-Spam-Level: ************************************************************
X-Spam-Status: Yes, score=57.834 tagged_above=2 required=6.31
tests=[AWL=10.800, BAYES_99=7.5, CHINANET=1, CNCGROUP=1.5, CNCJP=1.5,
DNSFRMRFCPST99=1.5, DNS_FROM_RFC_POST=0.1, DYN_DNSFRMRFCPST=3.5,
INVALIDYAHOOJP=1, INVYJP_DYN=3.5, ISO2022JP_BODY=-0.1,
NOTINCONTENTTYPE=0.2, QENCPTR1=0.2, REVDNSUNKNOWN=0.2,
SPF_SOFTFAIL=1.384, SURBL99=3.5, URIBL_AB_SURBL=3.812,
URIBL_JP_SURBL=1, URIBL_OB_SURBL=0.1, URIBL_SC_SURBL=4.498,
URIBL_WS_SURBL=2.14, URLBL_RBLJP=1.5, URLRBLJP99=2, URLRBLJP_DYN=5.5]
```

Fig.2 SpamAssasin Example 1 (SPAM)

```
X-Spam-Flag: YES
X-Spam-Score: 8.916
X-Spam-Level: ********
X-Spam-Status: Yes, score=8.916 tagged_above=2 required=6.31
tests=[AWL=-2.684, BAYES_99=7.5, DNSFRMRFCABS99=0.2,
DNS_FROM_RFC_ABUSE=0.1, NOTINCONTENTTYPE=0.2, SURBL99=3.5,
URIBL_OB_SURBL=0.1]
```

Fig.3 SpamAssasin Example 2 (HAM)

```
X-Spam-Flag: YES
X-Spam-Score: 7.814
X-Spam-Level: *******
X-Spam-Status: Yes, score=7.814 tagged_above=2 required=6.31
tests=[AWL=3.913, BAYES_50=0.001, CLICK_JP=1, CONTENT_TYPE_PRESENT=-0.1,
DNS_FROM_RFC_ABUSE=0.1, GEKIYASU=0.5, HAISHINTEISHI=0.3,
HIMITSUNO=0.1, HTML_MESSAGE=1, ISO2022JP_BODY=-0.1,
ISO2022JP_CHARSET=-0.1, MIME_HTML_ONLY=0.4, MURYOU=0.2, QENCPTR1=0.2,
RENRAKU=0.2, SUBJECT_ENCODED_TWICE=0.1, X_MAILER_PRESENT=0.1]
```

Fig.4 SpamAssasin Example 3 (HAM)

Fig.2 is the message judged at the score of 57.834 as Spam E-mail. However, the scores of examples as shown in Fig.3, and Fig.4 were 8.916 and 7.814 respectively which were judged as Spam, but the messages are Ham. Especially the score of "BAYES_99"' was 7.5 in Fig.2, and the result shows that classification capability of Bayesian filtering is not high. In this paper, 3,007 E-mails are accumulated in the database, where there are 2,913 Spams and 94 Hams misjudged as Spams.

3. Self Organizing Map(SOM)

The basic SOM can be visualized as a sheet-like neural network array as shown in Fig.5, the cells (or nodes) of which become specifically tuned to various input signal patterns or classes of patterns in an orderly fashion. The learning process is competitive and unsupervised, which means that no teacher is required to define the correct output for an input. Only one map node called a winner node at a time is activated corresponding to each input. The map consists of a regular grid of processing units. A model of some multidimensional observations, eventually a vector consisting of features, is associated with each unit. The map attempts to represent all the available observations with optimal accuracy using a restricted set of models. At the same time the models become ordered on the grid so that similar models are close to each other and dissimilar models are far from each other.

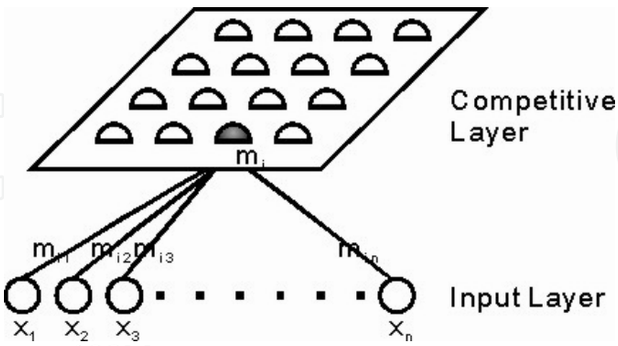

Fig.5 A basic architecture of SOM

Fitting of the model vectors is usually carried out by a sequential regression process. The n is the dimensioned number of input signals. An input vector $\mathbf{x}$ is compared with all the model vectors $\mathbf{m}_i(t)$. The best-match unit on the map is identified. The unit is called the winner.

For each sample $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$, first the winner index c (best match) is identified by the condition

$$\forall i, \|\mathbf{x} - \mathbf{m}_c\| \leq \|\mathbf{x} - \mathbf{m}_i\|$$

After that, all model vectors or a subset of them that belong to nodes centered around node c are updated at time t as

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(\mathbf{x}(t) - \mathbf{m}_i(t)) \quad for \forall i \in N_c(t)$$
$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) \quad otherwise$$

Here $h_{ci(x)}$ is the neighborhood function, a decreasing function of the distance between the i th and c th nodes on the map grid. The $N_c(t)$ specifies the neighborhood around the winner in the map array. This regression is usually reiterated over the available samples.
In this paper, we tried to classify the 3,007 E-mails into Spams and Hams by SOM in order to obtain the visual distribution intuitively.

4. Rule Extraction by ADG

4.1 Automatically Defined Groups

In the domain of data processing, to cluster the enormous data and to extract common characteristic from each clustered data are important for knowledge acquisition. In order to accomplish this task, we adopt a multi-agent approach, in which agents compete with one another for their share of the data, and each agent generates a rule for the assigned data; the former corresponds to the clustering of data, and the latter corresponds to the rule extraction in each cluster. As a result, all rules are extracted by multi-agent cooperation. However, we do not know how many rules subsist in given data and how data should be allotted to each agent. Moreover, as we prepare abundant agents, the number of tree structural program increases in an individual. Therefore, the search performance declines.

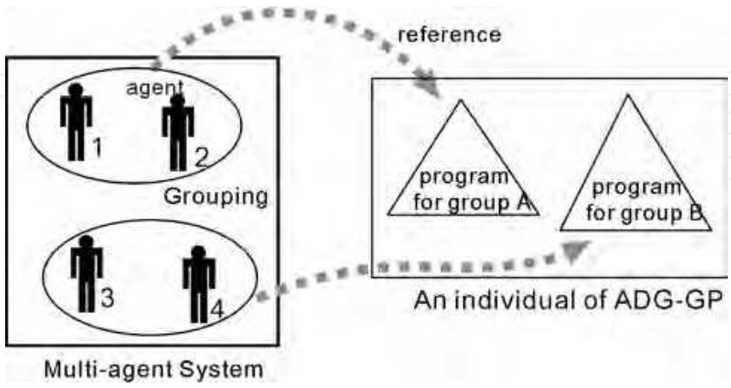

Fig.6 Concept of Automatically Defined Groups

In order to solve these problems, we have proposed an improved GP method, Automatically Defined Groups (ADG). The method optimizes both the grouping of agents

and the program of each group in the process of evolution. By grouping multiple agents, we can prevent the increases of search space and perform an efficient optimization. Moreover, we can easily analyze the behavior of agents. Respective groups play different roles from one another for cooperative problem solving. The acquired group structure is utilized for understanding how many roles are needed and which agents have the same role. That is, the following three points are automatically acquired by using ADG.

- How many groups (roles) are required to solve the problem?
- Which group does each agent belong to?
- What is the program of each group?

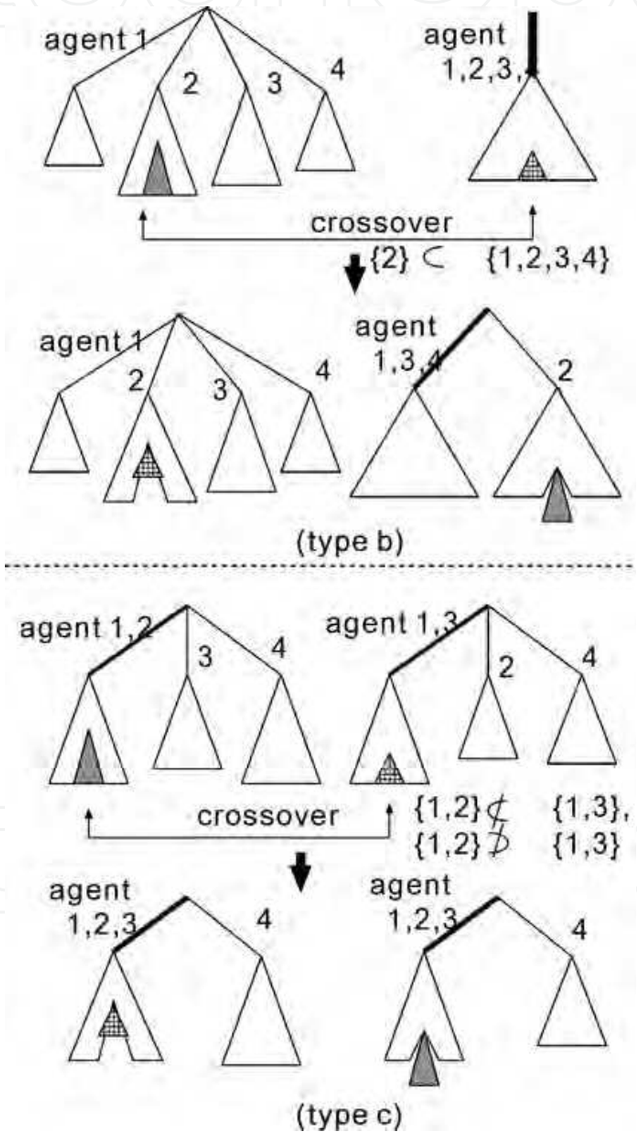

Fig.7 Examples of crossover

In ADG, each individual consists of the predefined number of agents. One GP individual maintains multiple trees, each of which functions as a specialized program for a distinct

group as shown in Fig.6. We define a group as the set of agents referring to the same tree for the determination of their actions. All agents belonging to the same group use the same program.

Generating an initial population, agents in each GP individual are divided into several groups at random. Crossover operations are restricted to corresponding tree pairs. For example, a tree referred to by an agent 1 in an individual breeds with a tree referred to by an agent 1 in another individual. In ADG, we also consider the sets of agents that refer to the trees used for the crossover. The group structure is optimized by dividing or unifying the groups according to the inclusion relationship of the sets.

The concrete processes are as follows: We arbitrarily choose an agent for two parental individuals. A tree referred to by the agent in each individual is used for crossover. We use T and T' as expressions of these trees, respectively. In each parental individual, we decide a set $A(T)$, the set of agents that refer to the selected tree T. When we perform a crossover operation on trees T and T', there are the following three cases.

(a) If the relationship of the sets is $A(T) = A(T')$, the structure of each individual is unchanged.

(b) If the relationship of the sets is $A(T) \supset A(T')$, the division of groups takes place in the individual with T, so that the only tree referred to by the agents in $A(T) \cap A(T')$ can be used for crossover. The individual which maintains T' is unchanged. Fig.7 (type b) indicates an example of this type of crossover.

(c) If the relationship of the sets is $A(T') \not\subset A(T)$ and $A(T) \not\subset A(T')$, the unification of groups takes place in both individuals so that the agents in $A(T) \cup A(T')$ can refer to an identical tree. Fig.7 (type c) shows an example of this crossover.

We expect that the search works efficiently and the adequate group structure is acquired by using this method.

4.2 Rule Extraction from classified data

In some kinds of databases, each data is classified into positive or negative case (or more than two categories). It is an important task to extract characteristics for a target class. However, even if data belong to the same class, all the data in the class do not necessarily have the same characteristic. A part of data set might show a different characteristic. It is possible to apply ADG to rule extraction from such classified data. In ADG, multiple tree structural rules are generated evolutionally, and each rule represents the characteristic of a subset in the same class data. Fig.8 shows a concept of rule extraction using ADG. Each agent group extracts a rule for the divided subset. The rules acquired by multiple groups can cover all the data in the target class.

We describe the detail of rule extraction from classified data. Here, the rules whether input data is classified into positive cases are extracted. In order to judge whether each data is regarded as positive case, we will find logical expressions such that the only positive data should satisfy.

Multiple trees in an individual of ADG represent the respective logical expressions. Each data in the training set is input to all trees in the individual. Then, calculations are performed to determine whether the data satisfy each logical expression. The input data is regarded as positive case if one or more logical expressions in the individual returns true. In

contrast, the input data is regarded as negative case if all logical expressions in the individual return false.

The concept of each agent's load arises from the viewpoint of cooperative problem solving by multiple agents. The load is calculated from the adopted frequency of each group's rule and the number of agents in each group. The adopted frequency of each rule is counted when the rule successfully returns true for each positive data. If multiple trees return true for a positive data, the tree with more agents is adopted. When the agent a belongs to the group g, the load of the agent w_a is defined as follows:

$$w_a = \frac{f_g}{n_{agent}^g}$$

where n_{agent}^g represents the number of agents which belong to the group g, and f_g represents the adopted frequency of g. By balancing every agent's load, more agents are allotted to the group that has a greater frequency of adoption. On the other hand, the number of agents in the less adopted group becomes small. Therefore, the number of agents of respective rules indicates how general each rule is for judgment of the class. Moreover, when some cases are judged to be true through a mistake of a rule, it is thought that the number of agents who support the rule should be small. To satisfy the requirements mentioned above, we will maximize the fitness f defined as follows:

$$f = -\frac{miss\_target\_data}{N_{Positive}} - \alpha \frac{misrecognition}{N_{Negative}} - \beta \frac{\sum_{N_{Negative}} fault\_agent}{misrecognition \times N_{agent}} - \delta V_w$$

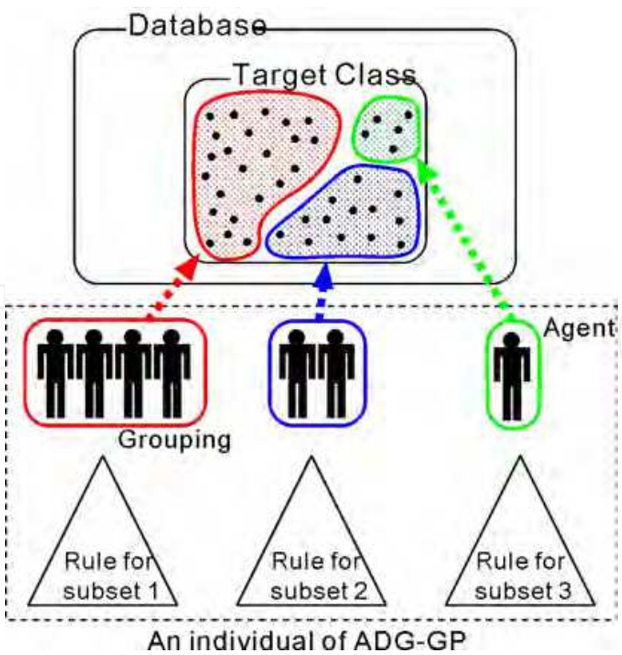

Fig. 8. Rule extraction using ADG

In this equation, $N_{Positive}$ and $N_{Negative}$ represent the number of positive cases and negative cases in database respectively. $miss\_target\_data$ is the number of missing data in the target positive data that should have been judged to be true. *misrecognition* is the number of mistakes through which negative data is regarded as positive case. When the rule returns true for negative data, $fault\_agent$ is the number of agents who support the wrong rule in each data. So, the third term represents the average rate of agents who support the wrong rules when misrecognition happens. By this term, the allotment of agents to a rule with more misrecognition will be restrained. V_w is the variance of every agent's load. By the fourth term, load balancing of agents will be achieved.
By evolution, one of the multiple trees learns to return true for positive cases, and all trees learn to return false for negative cases. Moreover, agents are allotted to respective rules according to the adopted frequency, and the allotment to a rule with more misrecognition is restrained. Therefore, the rule with more agents is the typical and reliable classification rule, and the rule with less agents is an exceptional rule for the rare case. This method is applied to the medical data and the effectiveness is verified (Hara, 2004, 2005).

5. Experimental Results

5.1 Learning for all cases and Extraction of rules(ichimura, 2007)

The map in the trained SOM as shown in Fig. 9 creates the distribution of Spam and Ham E-mails intuitively. The numerous label indicate the index of E-mails and each index is a category of E-mails which is classified the characters of judgment result by SpamAssassin. Each category usually has two or more classified E-mails. However, a category consisted of not only the message judged as Spam but the Ham E-mail misjudged as Spam. If the message is classified into this category, we have to reexamine the result of the classification, because the E-mail may be a Ham. In such a case, we will try to apply the sophisticate classification algorithm as shown in (ichimura, 2004; Ichimura, 2005; Yamaguchi 2008a; Yamaguchi 2008b).
Moreover, to make the process of reexamination automatically, we applied the extraction rules of finding a Ham misjudged as Spam by ADG. In ADG, two kinds of node are used for rule representation; function node and terminal node. In this experiment, the function node consists of $\{and, <, >\}$. The terminal nodes are two kinds of symbols; T_1, T_2. The set of T_1 is the name of rules defined in the SpamAssassin. The initial value of T_2 are given by a random number in $[0,1]$. In this paper, there are 300 individuals in GP. The number of agents in each GP is 50 respectively.
For the 3,007 E-mails judged as Spam by SpamAssassin, in which there were 2,913 Spams and 94 Hams misjudged as Spams, ADG can detect all of 94 Ham messages as No-Spam. In this experiment, the extracted rules to judge the Ham message correctly by ADG are shown in Fig.10. However, ADG could not judge correctly four messages of Spam as shown in Fig.11. That is, these four E-mails were misjudged as Ham, even if the correct judgment of messages is Spam. This shows excessive adaptation to the identification of Ham by ADG.

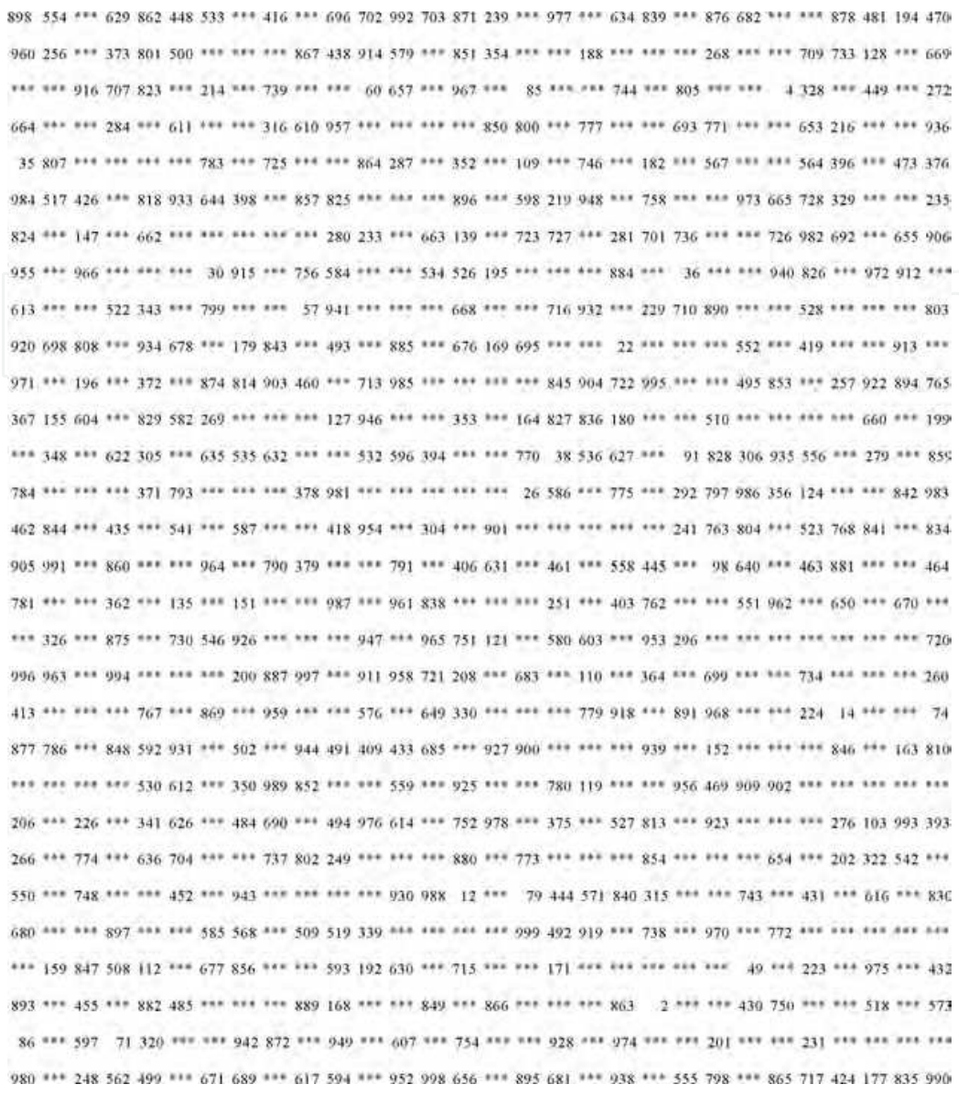

Fig. 9 Classification result of E-mails by SOM

```
Rule0 :
( ISO2022JP_BODY < -0.0257 ) and ( CHINANET < 0.7433 )
and ( URLRBLJP99 < 1.4865 ) and ( DYN_ONEGAI < 1.9846 )
and ( RCVD_IN_BL_SPAMCOP_NET < 0.1000 ) and ( URIBL_SC_SURBL < 4.4980 )

Rule1:
( BAYES_99 < 0.5507 ) and ( SHIFT_JIS2 < 0.1101 ) and ( DEETO < 0.0073 )
and ( USENBROAD < 0.0407 ) and ( RAZOR2_CF_RANGE_E4_51_100 < 0.1101 )
and ( UNDISC_RECIPS < 0.0618 ) and ( RIPE_NCC < 0.0997 )
and ( RCVD_IN_SBL < 0.0337 ) and ( URIBL_WS_SURBL < 0.1571 )
and ( FORGED_MUA_OUTLOOK < 0.2978 )
and ( HTML_OBFUSCATE_05_10 < 0.0073 ) and ( FUAN < 0.0147 )
and ( WINDOWS_CHARSET < 1.4665 ) and ( ADDRESS_IN_SUBJECT < 0.0391 )
and ( DREAMX < 1.4954 ) and ( RIMA_TDE_NET < 0.0734 )
and ( FORGED_RCVD_IP < 0.7343 ) and ( URIBL_OB_SURBL < 0.0073 )
```

Fig.10 Extracted rules by ADG

```
MisJudgement1
X-Spam-Flag: YES
X-Spam-Score: 9.215
X-Spam-Level: *********
X-Spam-Status: Yes, score=9.215 tagged_above=2 required=6.31
    tests=[ANATA=0.5, AWL=2.543, BAYES_50=0.001, CLICK_JP=1,
    CONTENT_TYPE_PRESENT=-0.1, DATE_IN_PAST_06_12=0.827, FORGED_RCVD_HELO=0.135,
    FROM_EXCESS_BASE64=1.309, FUAN=0.2, HAISHINTEISHI=0.3, HTML_MESSAGE=1,
    ISO2022JP_BODY=-0.1, MULTIPART_ALTERNATIVE=0.1, MURYOU=0.2, ONEGAI=0.2,
    PLING_QUERY=0.1, QENCPTR1=0.2, SUBJECT_ENCODED_TWICE=0.1, SUPPORT=0.1,
    TOOLONGSTR=0.5, X_MAILER_PRESENT=0.1]

MisJudgement2
X-Spam-Flag: YES
X-Spam-Score: 10.754
X-Spam-Level: **********
X-Spam-Status: Yes, score=10.754 tagged_above=2 required=6.31
    test=[AISHOU=0.2, ANATA=0.5, AWL=-3.046, BAYES_99=7.5, BUSINESS=0.2,
    CONTENT_TYPE_PRESENT=-0.1, HOTERU=0.5, ISO2022JP_BODY=-0.1,
    ISO2022JP_CHARSET=-0.1, MATTERU=0.3, MURYOU=0.2, OBSCURED_EMAIL=0.1,
    QENCPTR1=0.2, SUBJECT_ENCODED_TWICE=0.1, UNPARSEABLERELAY99=3.5,
    UNPARSEABLE_RELAY=0.5, UWAKI=0.3]
```

Fig.11 Misjudged Spam as Ham

5.2 Extraction of rules by using some trained SOMs

The precise classification of E-mails will require huge amounts of E-mails and the form of Spam will change into various ways every day. However, our method has a limitation in the amount of E-mails to extract classification knowledge once, because SOM cannot afford to train huge amounts of E-mails and ADG will be faced to a difficult judgment to extract a more important rule. Therefore, we improve our proposed method to use two or more SOMs simultaneously and then the portions of classification results by SOM are integrated into the terminals for rule extraction by ADG. Fig.12 shows the overview of our improved method by using some SOM modules and ADG. After scanning E-mails by SpamAssassin, the matching score of rules in the header of E-mail is recorded. As SpamAssassin has various rulesets described in the section 2 to detect a Spam, our improved method use two or more SOM modules which are trained for each ruleset. We can see some divided regions on the maps in each trained SOM. The part of rule falled into each divided region is given as the term such as $\{body0, body1, \cdots, header, \ldots, uri\}$ and then ADG maximizes the fitness f defined as follows:

$$f = -\frac{miss\_\text{target}\_data}{N_{\text{target}}} - \alpha \frac{misrecognition}{N_{non\_\text{target}}} - \beta \frac{\sum_{N_{non\_\text{target}}} fault\_agent}{misrecognition \times N_{agent}} - \delta V_w$$

In this paper, there are 300 individuals in GP. The number of agents in each GP is 100 respectively. The parameters of fitness functions are as follows: $\alpha = 1.0$, $\beta = 0.0001$, $\delta = 0.01$. ADG was trained for the random selected 500 cases.

We performed two kinds of experiments. In our experiment, the aim is to find E-mails with $score < 20$ from Spams. In the other experiments, the targets are E-mails with $score \geq 20$.

As the experimental result, we found that the correct ratio of classification capability by ADG was 84.0% if the total score of SpamAssassin is less than 20, otherwise about 86.6%. The sample of extracted rules are listed in the Table 1 and Table 2.

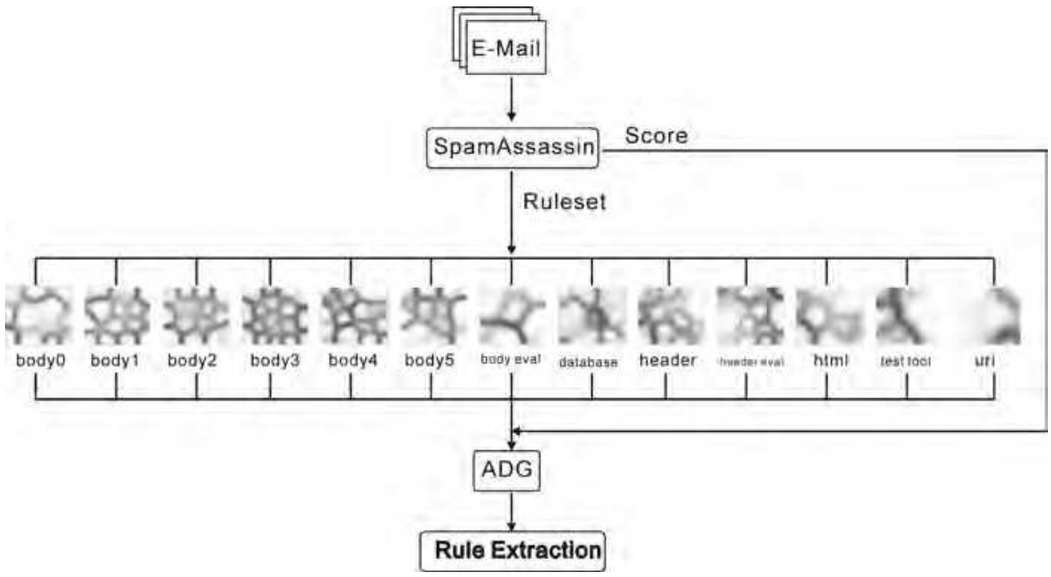

Fig.11 An overview of improved method by using some SOM modules

ID	#Agent	Rule
1	90	(and (eq header 4)(eq database 3))
2	5	(eq testtool 5)
3	4	(and (eq bodyeval 4)(and (eq header 8)(eq database 3)))
4	1	(and (eq header 3)(eq database 3))

Table 1. The judgement rule in case of $score < 20$

ID	#Agent	Rule
1	48	(and (eq testtool 2)(eq database 4))
2	14	(and (eq database 1)(eq testtool 2))
3	13	(and (eq database 2)(eq testtool 2))
4	7	(eq database 5)
5	3	(eq bodyeval 3)
6	2	(eq header 9)
7	2	(eq header 1)
8	2	(eq headereval 4)
9	2	(and (and (eq headereval 1)(eq uri 1))(eq bodyeval 1))
10	1	(eq html 4)
11	1	(eq html 2)
12	1	(eq header 10)
13	1	(eq headereval 3)
14	1	(eq header 2)
15	1	(eq body3 12)
16	1	(eq testtool 1)

Table 2. The judgement rule in case of $score \geq 20$

6. Conclusion

In this paper, we propose a classification method for Spam E-mail based on the results of SpamAssassin. This method can learn patterns of Ham and Spam E-mails. First, SOM can classify many E-mails into the some categories. In this phase, we can see the characters of current received Spam E-mails. Second, ADG can extract the correct judgment rules of Hams misjudged as Spams. However, there are a few cases of Spam misjudged as Ham. In this experiment, ADG makes an over fitting to the characters of Hams. We have met the problems according to the limitation of classification capability by SOM and explosive search in GP using many nodes as shown in T_1 Therefore, we improve the proposed method to classify the Spam E-mails by using some SOM modules and to extract some rules by ADG according to the classification results.

Moreover, these methods give the input patterns using the judgment rules in SpamAssassin and their agreement value. Even if one rule is fire and the other rules are not fire, and the agreement value is high, the result of judgment will be Spam. The SpamAssassin is very useful open source software, the learning tool is equipped in itself. But we may meet the misjudgment. The rule must be redefined by each user according to user's work style and so on. In order to improve such problems, the natural language processing and the mining of important word from E-mails will be required. We will develop the hybrid classification and rule extraction method using the learning of neural networks and ADG.

7. References

The Apache SpamAssassin Project. SpamAssassin http://spamassassin.apache.org/

Kohonen, T.(1995). *Self-Organizing Maps*, Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, New York

Hara, A. & Nagao, T. (1999). Emergence of cooperative behavior using ADG; Automatically Defined Groups, Proc. of the 1999 Genetic and Evolutionary Computation Conf., pp.1039-1046

Hara, A.; Ichimura, T.; Takahama, T. & Isomichi, Y. (2004). Discovery of Cluster Structure and The Clustering Rules from Medical Database Using ADG; Automatically Defined Groups, In *Knowledge-Based Intelligent Systems for Healthcare*, Ichimura, T. & Yoshida, K. (Ed.), Advanced Information and Knowledge Processing, pp.51-86

Hara, A.; Ichimura, T.; & Yoshida K. (2005). Discovering multiple diagnostic rules from coronary heart disease database using automatically defined groups, *Journal of Intelligent Manufacturing*, Vol.16, No.6, pp.645-661

Hara, A.; Kurosawa, Y.; Ichimura, T. (2008). Automatically Defined Groups for Knowledge Acquisition from Computer Logs and Its Extension for Adaptive Agent Size, In *Current Trends in Intelligent Systems and Computer Engineering*, Springer, pp.15-32

Frederick P. Brooks, Jr. The Postfix Home Page, http://www.postfix.org/

Amavisd-new. The amavisd-new Web site, http://www.ijs.si/software/amavisd/

Clam AntiVirus. The Clam AntiVirus Web site, http://www.clamav.net/

Ichimura, T.; Oeda, S.; Suka, M. & Yoshida, K. (2004). A learning method of immune multi-agent neural networks, *Neural Computing and Applications Journal*, Vol.14, pp.132-148

Ichimura, T.; Oeda, S.; Suka, M.;Hara, A.; Mackin, K.J. & Yoshida, K. (2005). Knowledge Discovery and Data Mining in Medicine, In *Advanced Techniques in Knowledge Discovery and Data Mining(Advanced Information and Knowledge Processing)*, Nikhil, P.; Jain, L.C. Ed., Springer, pp.177-210

Yamaguchi, T.; Ichimura, T.; & Mackin, K.J. (2008). Adaptive Tree Structured Clustering Method using Self-Organizing Map, *Proc. of Joint 4th International Conference on Soft Computing and Intelligent Systems and 9th International Symposium on advanced Intelligent Systems (SCIS & ISIS 2008)*, pp.1407-1411

Yamaguchi, T.; Ichimura, T.; & Mackin, K.J. (2008). Analysis using adaptive tree structured clustering method for medical data of patients with coronary heart disease, *Proc. of 4th International Workshop on Computational Intelligence and Applications 2008 (IWCIA 2008)*, pp.139-144

Ichimura, T.; Kurosawa, Y. & Hara, A. (2007). A classification Method for Spam E-mail by Self-Orgaizing Map and Automatically Defined Groups, *Proc. of the 2007 IEEE Intl. Conf. on System, Man and Cybernetics (SMC2007)*, pp.2044-2049

Self-Organizing Maps

Edited by George K Matsopoulos

ISBN 978-953-307-074-2
Hard cover, 430 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010

The Self-Organizing Map (SOM) is a neural network algorithm, which uses a competitive learning technique to train itself in an unsupervised manner. SOMs are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space and they have been used to create an ordered representation of multi-dimensional data which simplifies complexity and reveals meaningful relationships. Prof. T. Kohonen in the early 1980s first established the relevant theory and explored possible applications of SOMs. Since then, a number of theoretical and practical applications of SOMs have been reported including clustering, prediction, data representation, classification, visualization, etc. This book was prompted by the desire to bring together some of the more recent theoretical and practical developments on SOMs and to provide the background for future developments in promising directions. The book comprises of 25 Chapters which can be categorized into three broad areas: methodology, visualization and practical applications.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen