

A False Positive Safe Neural Network

The Followers of the Anatrium Waves

Alexandru Catalin COSOI

Senior Researcher / BitDefender AntiSpam Laboratory

<mailto:acosoi@bitdefender.com>

Abstract: Content-based filters (e.g. Keyword Filters, Heuristics Filters, Statistical Learning Filters, Pattern Recognition Neural Networks, and so on) use tokens, which are found during message content analysis, to separate spam from legitimate messages. The effectiveness of these token-based filters is due to the presence of token signatures (e.g. tokens that are invariant for the many variants of spam messages). As many scientific researchers in this field might have noticed, a new trend of spam messages appeared, that have a low frequency of token signatures, thus making them significantly more difficult to identify. What once had variations in just a part of the message, new formulations can be seen now on the entire message. We believe that good old content based filters can still do a pretty good job, if they are trained accordingly. Also, the most important part of the paper is represented by an add-on which can be brought to any type of neural networks in order to minimize false positives, consisting in an extra set of relevancies assigned to individual features.

Key Words: ART, ARTMAP, Spam, AntiSpam, Heuristics, Feature Extraction, Anatrium

INTRODUCTION

The currently employed infrastructure for eMail transfer, the simple mail transfer protocol (SMTP), hardly provides any support for detecting or preventing spam. We are also lacking a widely accepted and deployed authentication mechanism for sending emails. Thus, until a new unlikely global email infrastructure will be developed so as to allow a better control of this problem, there are two current major approaches that show the greatest potential for coping with the problem: detecting spam based on content filtering or preventing spam to enter our mailboxes by using

techniques such as reputation management, white-listing, increasing the costs associated with sending out email messages, and so on.

Current Token-based spam filters (e.g. Signature Filters, Heuristic Filters, Neural Network Filters, Bayesian Filters, Support Vector Machines, N-gram, TF-IDF, LSA – latent semantic analysis, contextual network graphs, and so on) distinguish between spam and legitimate email messages based primarily on the tokens found in those messages' text. However this approach has had mixed results. On one hand, many spam messages have token signatures that facilitate filtering. These signatures typically consist of tokens that are invariant for many variants automatically generated by spammers. On the other hand, spammers can use various techniques to defeat this filters. (Pu et. all, 2006)

Judging by the frequency of their updates, we noticed that token-based filters can be classified in two major categories:

1. Long term filters (updated weekly or monthly, or maybe early)
2. Short term filters (updated hourly or daily, or at the most, weekly)

Each of these filters makes use of a feature/token extraction algorithm in order to have enough information for a good spam vs. legitimate classification. It is also known that long term filters have incredible good detection rates in laboratory conditions, while short term filters have registered very good detection rates in real world conditions.

THE PROBLEM

The only characteristic of spam messages that has resisted during its evolution, is the fact that they are launched in waves. A few years ago, a single message

was multiplied a few hundred thousand times and sent to a large database of email addresses; the message usually changed after a few days. Different solutions were found for that problem (heuristic filters, Bayesian filters, etc), and spam had to mutate into something new.

The new wave types contained messages that were unique computational speaking, by having inserted random [legitimate] text. Usually, spammers were not making any effort in modifying the inter-wave spam part of the message. For that specific situation, changing Bayes poison and the URL was enough to fool most of the anti-spam solutions. The spam part of the message also was changed daily. *Nowadays, waves contain totally different messages: we can see entire spam waves, with just a single characteristic in common: its structure, and even this is only temporary.*(Musat, 2006)

Among the first waves of spam messages which begun meeting these new characteristics was the one that advertised Anatrium. We focused our attention on it, because although it was an easy target, rarely changing its body text, it changed its subject very often; so often that in a single wave, just a few samples had the same subject.

2007 was its year of glory, and then we observed that each time there was a different subject and also they were changing the title of the email in almost any situation. The rest of the body also changed but less often. After gathering messages for about a month, we managed to notice that there is a inter wave pattern that cyclically repeats. There were about 40 rewords for the subject and around 50 for the "title" of the message. Considering only the subject and title, there are almost 200 variations. By extrapolating, we came up with a simple but very powerful spam script, that we think lies (or it will lie, since more and more spam waves follow this structure) at the base of any spammer's bag of tricks.

Databases:

- D : Random legitimate text
- D_1 : Different formulations of a certain spam phrase
- D_2 : Different formulations of another spam phrase
-
- D_n : Different formulations of another spam phrase

Create spam message script:

1. Choose a random phrase from D_1
2. Choose random text from D
3. Choose a random phrase from D_2
4. Choose random text from D
5.
6. Chose random phrase from D_n

Send message.

And then we noticed the problem. Let's say an automated feature extraction algorithm would start looking for patterns in the *spam waves* received in the previous hours. On the Anatrium wave, it will luckily find a few words that appear in all the messages, but most probably, it won't find anything in common between subsequent messages.

Based on our research, training either a bayesian filter *on the similar (e.g. same structure/layout) messages received in one day*, or using a pattern discovery algorithm (ex: Teiresias alg.), we will obtain outcomes regarding only the volatile part of the spam messages (like Bayes poison, or random spam message information) and not the important spam paragraphs (e.g. titles and subject in the case of the Anatrium wave).

That is to say that training our filters on this wave of email messages would offer us a pretty good detection rate but with the downfalls of higher memory usage (*many individual patterns would have to be kept in memory*) and a short lifetime of the detection features.

Below, find a list of samples of subjects for this particular type of spam:

- Less weight - more pleasure and joy
- Watch the pounds disappear
- Healthy living with less fat
- Can you imagine that you are healthy
- Shed weight now and enjoy the process
- NOW save on meds you need
- Say goodbye to extra pounds
- Losing weight has never been so easy

If you Google™ any of these subjects, you will find thousands of blog comments/spam advertising Anatrium, following the same pattern.

Also, it is thought, that *this script appeared as a natural consequence of the bot armies*. If each infected computer sends just one sample to a few million users, since a spammer can rent or buy nowadays a few hundred computers, he will achieve similar results with this script (of course, much better since these are machines spread all around the world)

PROPOSED METHOD

There are two major approaches suitable for dealing with this problem:

1. *On-line cumulative training*, which means that if a system would learn each time something new arrives, since the ways a phrase can be rephrased are not infinite, after a certain amount of time, if the features

aren't too exclusive (but rather weak features) the system will be able to correctly recognize as spam a certain email even though that email was never seen before (e.g. self organizing neural networks, or Bayesian filters, mainly machine learning filters). This method can be performed either on the client side, which means that it will require a lot of user feedback (which rarely happens), or *on the vendor side*, which could mean *either a high frequency of the updates, or a lower frequency but larger data files*.

2. *Monthly batch training*, or at least on longer periods of time, *in order to extract those cyclical features that could guarantee a proactive detection of the future spam waves*.

Of course, those two techniques can also be combined. The basic idea was that *we need to extract features not from a daily corpus, but rather from a corpus two months behind, and preferably not exclusive ones*. A good way to create strong patterns would be to use a neural network that combines short weaker patterns (if the email has words like "Viagra", "Valium", or if the date of the message is in the future and so on), which individually have a high false positive rate, in order to create large strong patterns.

A suitable neural network type up for this task would be ARTMAP networks (Cosoì, 2006). ARTMAP architectures are neural networks that develop stable recognition codes in real time into response to arbitrary sequences of input patterns. They were designed to solve the stability-plasticity dilemma that every intelligent machine learning system is facing: *how to keep learning from new events without forgetting previously learned information*. ARTMAP networks were designed to accept binary or fuzzy input patterns (Carpenter & Grossberg, 1991). ARTMAP networks consist of two ART1 networks, ARTa and ARTb, bridged via an inter-ART module, as shown on

Adaptive Resonance Theory (ART) was developed by Carpenter and Grossberg over the period of 1976-86, during their studies of the behavior of models of systems of neurons. The couple has published a large number of papers, in which they develop the mathematical theory of such models, usually in terms of systems of differential equations, and they have applied this to actual neural systems. Like Kohonen, they have been particularly interested in systems that are capable of organizing themselves. The ART paradigm can be described as a type of incremental clustering. It has the ability to learn without supervised training and is consistent with

cognitive and behavioral models. It is an unsupervised paradigm based on competitive learning which is capable of automatically finding categories and creating new ones when they are needed. As such, it is a close model of the prototype theory of concept attainment.

The ART1 system consists of an attentional subsystem and an orienting subsystem as shown in figure 1. The attentional subsystem consists of two competitive networks, the comparison layer *F1* and the recognition layer *F2*, and two control gains, Gain 1 and Gain 2. The orienting subsystem contains the reset layer for controlling the attentional subsystem overall dynamics. The comparison layer receives the binary external input passing it to the recognition layer responsible for matching it to a classification category. This result is passed back to the comparison layer to find out if the category matches that of the input vector. If there is a match a new input vector is read and the cycle starts again. If there is a mismatch the orienting system is in charge of inhibiting the previous category in order to get a new category match in the recognition layer. The two gains control the activity of the recognition and comparison layer, respectively.

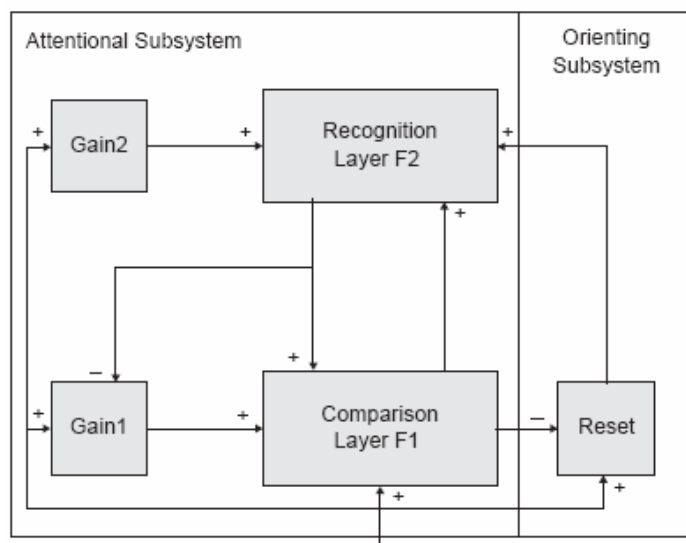


Figure 1 – ART Architecture

The weight vectors are initialized to 1, for instance, $w_{j1}=1, w_{j2}=1, \dots, w_{jD}=1, 1 \leq j \leq C$. The orienting subsystem is a qualifier, where the match function of the candidate elected by the attentional system is compared against the vigilance ρ . It uses the winnertake-all learning strategy. If the condition satisfied, the pattern will be learned by the winning node, otherwise the activation function for the candidate will be set to 0, and a new candidate is

ected and tested. The searching procedure keeps on going until either a candidate meets the vigilance constraint or no more candidates are left. If none of the output nodes can encode the pattern, a new node is committed to the pattern.

The goal of the network training is to find a set of templates, which best represent the underlying structure of the samples. Suppose the set of templates $W = \{w_1, w_2, \dots, w_n\}$ and the number of patterns from X associated with each template $N = \{N_1, N_2, \dots, N_C\}$. It is important to note that the number of output nodes, sets W and N are growing dynamically, which is fact the beauty of ART networks.

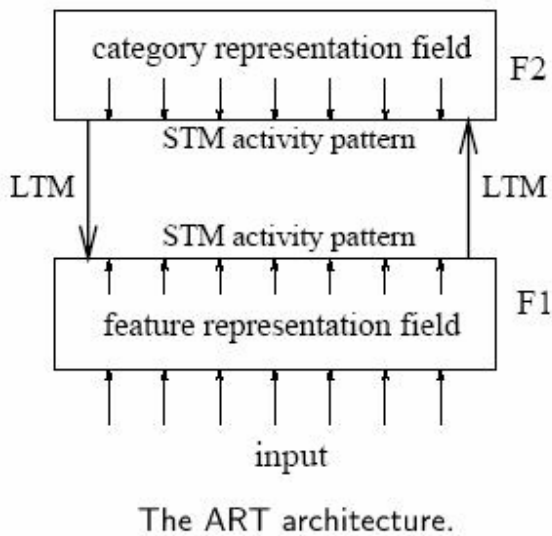


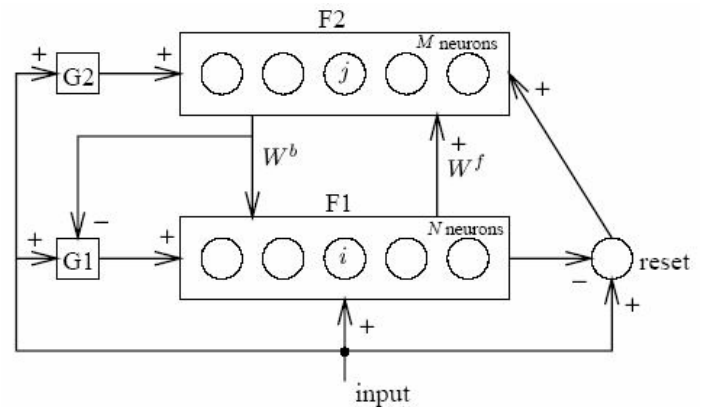
Figure 2 – Schematic ART Architecture

Before learning can change memories, ART treats the chosen code as a hypothesis, which it tests by matching the top-down expectation of y against the input that selected it. Parallel specific and nonspecific feedback from F2 implements matching as a real-time locally defined network computation. Nodes at F1 receive both learned excitatory signals and unlearned inhibitory signals from F2. These complementary signals act to suppress those portions of the pattern I of bottom-up inputs that are not matched by the pattern V of top-down expectations. The residual activity x^* represents a pattern of critical features in the current input with respect to the chosen code y . If y has never been active before, $x^*=x=I$, and F1 registers a perfect match.

If the matched pattern x^* is close enough to the input I , then the memory trace of the active F2 code converges toward x^* . The property of encoding an attentional focus of critical features is key to code stability. This learning strategy differentiates ART networks from MLPs, which typically encode the

current input, rather than a matched pattern, and hence employ slow learning across many input trials to avoid catastrophic forgetting. ART memory search begins when the network determines that the bottom-up input I is too novel, or unexpected, with respect to the active code to satisfy a matching criterion. The search process resets the F2 code y before an erroneous association to x^* can form. After reset, medium-term memory within the F1→F2 pathways (Carpenter and Grossberg, 1990) biases the network against the previously chosen node, so that a new code y^* may be chosen and tested.

The ART matching criterion is determined by a parameter ρ called vigilance, specifies the minimum fraction of the input that must remain in the matched pattern in order for resonance to occur. Low vigilance allows broad generalization, coarse categories, and abstract memories. High vigilance leads to narrow generalization, fine categories, and detailed memories. At maximal vigilance, category learning reduces to exemplar learning. While vigilance is a free parameter in unsupervised ART networks, in supervised networks vigilance becomes an internally controlled variable which triggers search after rising in response to a predictive error.



The ART1 neural network.

Figure 3 – Functionality and weights

ARTMAP employs a preprocessing step called *complement coding*, which, by normalizing input patterns, solves a potential category proliferation problem (Carpenter, Grossberg, and Rosen, 1991). Complement coding doubles the number of input components, presenting to the network both the original feature vector and its complement. In neurobiological terms, complement coding uses both on-cells and off-cells to represent an input pattern. The corresponding on-cell portion of a weight vector encodes features that are consistently present in category exemplars, while the offcell portion encodes features that are consistently absent. Small weights in complementary portions of a

category representation encode as uninformative those features that are sometimes present and sometimes absent.

A central feature of all ART systems is a pattern matching process that compares an external input with the internal memory of an active code. ART matching leads either to a *resonant* state, which persists long enough to permit learning, or to a parallel memory search. If the search ends at an established code, the memory representation may either remain the same or incorporate new information from matched portions of the current input. If the search ends at a new code, the memory representation learns the current input. This *match-based learning* process is the foundation of ART code stability. Match-based learning allows memories to change only when input from the external world is close enough to internal expectations, or when something completely new occurs. This feature makes ART systems well suited to problems that require online learning of large and evolving databases.

In the training phase, the system has to receive a list of features extracted from the email messages and an output category. For example, ARTa will receive an input vector where each field indicates the existence of a certain spam or legitimate characteristic. Also, each input vector will be associated to a label which indicates if the current pattern was extracted from a spam or a legitimate email message, which will be fed to the ARTb module. When the training phase starts, the system will quickly associate inputs and outputs by creating strong patterns for each category.

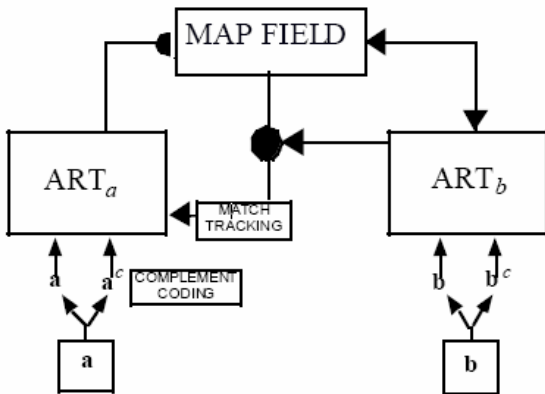


Figure 4 – ARTMAP representation

The results are very good (Cosoi, 2006), with a false positive rate of almost 1% (which is not exactly the best yet obtained, but it can be rated among the top 6 AntiSpam filters – this test was made in 2006) and a false negatives rate under 10%. The problem that appears is that since the training phase is performed on a few million legitimate and spam messages samples,

and since the individual heuristics are generally weak, the extracted patterns can be quite confusing for the neural network algorithm. For example we can have a situation where important legitimate features and standard weak spam features can determine a mistakenly “this is spam” answer, and vice-versa.

These situations are generally determined by the large corpus of messages on which the neural network has to train in order to achieve an acceptable accuracy. *In many situations, in our experiments, the training phase stopped after a fixed number of training iterations was achieved, and not when reaching a pre-established accuracy* – hence the false positives rate was high.

The solution we found to address this problem was to offer an a priori numerical relevance to each individual feature, and also to the category (spam or legitimate) for which this feature was created. Our purpose was to create an inhibited connection, in order to stop the neural network from giving an answer if the relevance of the pattern was smaller than a pre-established threshold T. Of course, this means that good hits would be eliminated to, but common-sense would say that we can’t actually say an email is a spam message only because it contains the word “Viagra”.

If we consider I and S the relevance for the legitimate heuristics within a subset of a pattern and respectively S the relevance for the spam heuristics, we can combine them in a total relevance for a pattern by using the following simple rule:

$$R = \frac{1 - H + S}{2} \quad (1)$$

Where, H and S are computed as percentages of the total sum of the relevancies within a pattern (e.g. the sum of individual relevancies divided by the sum of the maximal relevancies these heuristics can achieve).

By using this result, the neural network can determine if this is an important pattern for the decision process or not. *Consequently, this approach is more similar to a heuristic filter than to a neural network. In order to keep all the facilities that a neural network would offer, (and we also chose this type of neural network in order to solve the stability-plasticity dilemma), we had to add a punishment-reward system in the control subsystem of the ARTa module.* The process we developed is quite simple to explain. Each time the prediction matched the expectation we increased by a small amount the relevance of that pattern. If the prediction and the expectation were different, we decreased the relevance with a small

amount. The process can be defined using the following formula.

$$R_{i+1} = (1-w)R_i + w(R + (-1)^c \cdot \frac{w}{100}) \quad (2)$$

Where $(-1)^c$ has a negative value when the expectation and the prediction are different, and a positive one when the two are the same.

$$Score = \alpha_1 ScorNN + \alpha_2 (\alpha_1' Rel_1 + \alpha_2' Rel_2) \quad (3)$$

$Scor$ represents the spaminess, $ScorNN$ represents the score returned by the Neural Network, Rel_1 , represents the *relevance of the pattern created during training*, and Rel_2 represents the relevance of the pattern computed with $R = \frac{1-H+S}{2}$

$$Rel_2 = \frac{1-H+S}{2} \quad (4)$$

We use this relevance for fast emergency updates. If problems appear with a feature, by decreasing its relevance, it will decrease its pattern relevance.

$$Rel_{1(0)} = \frac{1-H+S}{2} \text{ (just as initial value)} \quad (5)$$

The initial value is computed from the feature relevancies, but it will be adapted by the punishment reward system. All the α 's are pre-computed weights for each term.

TRAINING PHASE

If $((Score \geq Threshold) \ \&\& \text{ (given category equals output category)})$

$$Rel_{1(x)} + = \varepsilon$$

Update weights (basic neural network behavior)

Else

$$Rel_{1(x)} - = \varepsilon$$

Update weights (basic neural network behavior)

TESTING PHASE

If $((Score = \alpha_1 ScorNN + \alpha_2 (\alpha_1' Rel_1 + \alpha_2' Rel_2)) > \text{Predetermined Threshold})$

Print OutputCategory

Else

Don't say anything

As it can be seen, this system is actually a *hybrid between a neural network* (which creates patterns from simple features) *and a heuristic filter*, which prevents weak patterns from taking part in the decision process. This way, we can solve the Anatrium wave by using weak features (like the word “Anatrium” for instance, which in this case is obvious, but also “lover” and “extra pounds” will become powerful features, without the risk of false positives by using a low relevance and as a feature in a larger pattern.

RESULTS

Our *laboratory* tests showed that by applying the improvements presented in this paper, the false positives rates dropped radically from an initial 1% to 0.0001%, while the false negative rate reached 20%, (in time this rate increases consistently without at least monthly training of the neural network) compared to an initial value of 3% (Cosoi, 2006).

The conditions in which the experiments took place are the following:

- 2.5 million spam messages (sampled on waves with a high degree of variation) and around 1000 simple low relevance text heuristics (not counting the standard header heuristics). A good method to reproduce this experiment is to take the first 1000 words (ordered by discrimination, but with a minimum of 10-30 hundred occurrences) from a bayesian dictionary trained on this corpus, and also standard header heuristics.
- Almost 1 million legitimate email messages
- 75% of the message corpus were used for training the neural network and,
- 25% were used in testing the neural network.

We also performed a test on the TREC2006 spam corpus. After eliminating the spam messages which contained images, unreadable charsets and malformed messages, we obtained a false negative rate of 45%, and 2 false positives. The reason for this low detection rate is the fact that the individual features were created for more recent spam. We also tried training on a part of this corpus (standard 75% training and 25% testing), but that only increased our detection rate with 10%. (65% in total). We are confident though, that extracting a few features from this specific timeframe, would consistently increase our detection

rate even on this corpus.

CONCLUSIONS

The spam script presented in this paper might cause AntiSpam vendors a few problems, but these are problems which don't require a huge amount of code to solve, but rather a change of perception. If the future of spam consists in randomizing (by a certain amount of course) even those phrases that contain the spam message itself (not only Bayes Poison), a good solution for this problem would be to perform analysis on a larger pool of spam messages (e.g. representative samples from a larger timeframe). Also, we might notice that old content based filters (e.g. Bayesian filters or Neural Network Filters) can still do a pretty good job.

The Neural Network model (e.g. the modified ARTMAP model) presented here, can be considered a good asset to any AntiSpam solution, because of its proactive detection using non-exclusive heuristics and its extremely low false positive rate. Also, the inhibitory connections and the modified learning phase can also be used for other neural network models.

REFERENCES

1. Pu C., Webb S., Kolesnikov O., Lee W., Lipton R. (2006). "Towards the Integration of Diverse Spam Filtering Techniques" - IEEE International Conference on Granular Computing
2. Beckman S. (2007). High-Performance Asynchronous IO for SMTP Multiplexing, SpamConference, Boston MIT
3. Cosoi A. C. (2006). An AntiSpam filter based on adaptive neural networks, SpamConference, Boston MIT
4. Graham P. (2002). A plan for spam
5. Carpenter, G. & Grossberg, S. (1991). Supervised real-time learning and classification of non-stationary data by a self-organizing neural network, In: *Pattern recognition by self organizing neural networks*, Carpenter, G. & Grossberg, S., (Ed. MIT press), 501-544, Publisher MIT press, ISBN 0-262-03176-0, Cambridge Massachusetts
6. Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization—Papers from the AAAI Workshop*, pp. 55-62, Madison Wisconsin. AAAI Technical Report WS-98-05, 1998
7. Androutsopoulos, I., Koutsias, J., Chandrinou, K., Paliouras, G., Spyropoulos, C. An Evaluation of naïve Bayesian Anti-Spam Filtering. In *Proceedings of the workshop on Machine Learning in the New Information Age*, 11th European Conference on Machine Learning, pp 9-17, Barcelona, Spain, 2000
8. Rich Drewes, An artificial neural network spam classifier, CS676, Prof. Sushil Louis, 2002
9. Daniel Kelleher, Spam Filtering Using Contextual Network Graphs, The university of dublin, Department of computer Science
10. Shlomo Hershkop, Behavior-based Email Analysis with Application to Spam Detection, Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Graduate School of Arts and Sciences, Columbia University
11. Musat Claudiu, Layout Based Spam Filtering, proceedings of world academy of science, engineering and technology volume 12 march 2006 issn 1307-6884
12. Jingrui He, Bo Thiesson, Asymmetric Gradient Boosting with Application to Spam Filtering, CEAS 2007
13. D. Sculley, Online Active Learning Methods for Fast Label-Efficient Spam Filtering, CEAS 2007
14. Cosoi Catalin, Assigning Relevancies To Individual Features For Large Patterns In Artmap Networks, DAAAM 2007 (ISI Proceedings, Danube Adria Association for Automation and Manufacturing)
15. Calton Pu, Steve Webb, Observed Trends in Spam Construction Techniques: A Case Study of Spam Evolution, CEAS 2006
16. Ah-Hwee Tan "Cascade ARTMAP: Integrating Neural Computation and Symbolic Knowledge Processing", IEEE Transactions on Neural Networks journal, 1997
17. Ah-Hwee Tan "Adaptive Resonance Associative Map: A hierarchical ART system for fast stable associative learning", Proceedings of IJCNN, 1992
18. Gail A. Carpenter and Stephen Grossberg, ADAPTIVE RESONANCE THEORY, Department of Cognitive and Neural Systems, Boston University
19. T. Tanaka and A. Weitzenfeld - Adaptive Resonance Theory