An Interactive Hybrid System for Identifying and Filtering Unsolicited E-mail

M. Dolores del Castillo and J. Ignacio Serrano

Instituto de Automática Industrial, CSIC, Ctra. Campo Real km 0.200 – La Poveda, 28500 Arganda del Rey, Madrid. Spain
{lola, nachosm}@iai.csic.es

Abstract. This paper presents a system for automatically detecting and filtering unsolicited electronic messages. The underlying hybrid filtering method is based on e-mail origin and content. The system classifies each of the three parts of e-mails separately by using a sinole Bayesian filter together with a heuristic knowledge base. The system extracts heuristic knowledge from a set of labelled words as the basis on which to begin filtering instead of conducting a training stage using a historic body of pre-classified e-mails. The classification resulting from each part is then integrated to achieve optimum effectiveness. The heuristic knowledge base allows the system to carry out intelligent management of the increase in filter vocabularies and thus ensures efficient classification. The system is dynamic and interactive and the role of the user is essential to keep the evolution of the system up to date by incremental machine learning with the evolution of spam. The user can interact with the system over a customized, friendly interface, in real time or at intervals of the user's choosing.

Keywords: e-mail classification, machine learning, heuristic knowledge.

1 Introduction

Unsolicited commercial e-mail, known as "spam", is widely recognized as one of the most significant problems facing the Internet today. According to a report [5] from the Commission of European Communities, more than 25% of all e-mail currently received is spam. More recent reliable data indicate that this percentage has increased by over 50%.

There are techniques for preventing addresses from being discovered and used by spammers, such as encoding or hiding the kinds of data that spammers target. Unfortunately, these techniques are not in widespread use [21]. Since spam growth is exponential and prevention is both extremely difficult and rare, the problem must be tackled on a technical front, by developing methods to analyse e-mail traffic in order to identify and reject spam communications. This introduction provides a review of the properties of the different filter types that are currently available.

Filtering can be classified into two categories, origin-based filtering or content-based filtering, depending on the part of the message chosen by the filter as the focus for deciding whether e-mail messages are valid or illegitimate [6]. Origin-based filtering focuses on the source of the e-mail, which is recorded in the domain name and address of the sender device. Two types of origin-based filters are available [12]:

E. Corchado et al. (Eds.): IDEAL 2006, LNCS 4224, pp. 779 – 788, 2006.
© Springer-Verlag Berlin Heidelberg 2006

White-list filtering. This kind of filtering only lets e-mail from explicitly confidential or reliable lists of e-mail addresses (white lists) through. TDMA [20] and ChoiceMail [16] are paradigms of white-list filtering.
Black-list filtering. These filters use lists of e-mail addresses that are widely known to be spam sources (black lists). Razor [23] and Pyzor [15] are tools that use black-list filtering.

Content-based filters conduct an analysis whose goal is to review the text content of e-mail messages. Depending on the analysis technique used, these filters may be differentiated as follows [12]:

Rule-based filters. This type of filter extracts text patterns or rules [4] and assigns a score to each rule based on the occurrence frequency of the rule in spam and non-spam e-mail in a historic body of e-mail. SpamAssassin is the most popular application using rule-based filtering [18].
Bayesian filters. These filters analyse every word of a message and assign a spam probability and a non-spam probability to each word based on statistical measurements. Next, the Bayes theorem is used to compute the message total probability [1], and the message is categorized according to the higher probability value. There are a great many filters that use these properties [10], [11].
Memory-based filters. These filters use e-mail comparison as their basis for analysis. E-mail messages are represented as feature or word vectors that are stored and matched with every new incoming message. Some examples of this kind of filter are included in [2], [7]. In [6], case-based reasoning techniques are used.
Other filters. Some of the content-based approaches adopted do not fall under the previous categories. Of these, the most noteworthy are the ones based on support vector machines [9] or neural networks [22].

Other features exist that can be used to classify filters in other ways as well. These features include filter location (in a client or dedicated server), ease of use, configuration ability, and filtering options. One of the basic properties of filters is their dynamism or ability to evolve over time. Only some filters have this ability to learn from past mistakes. Bayesian filters evolve by updating word probabilities and including new words in their vocabularies, while memory-based filters evolve by increasing the number of stored e-mail messages. Filters that rely on lists updated by users are also dynamic. Other filtering systems, such as some rule-based filters, are static and once such filters have been installed, their behaviour and performance never change.

Most current filters achieve an acceptable level of performance, detecting 80%-98% of spam. The main difficulty is to detect false positives, i.e., the messages that are misidentified as spam. Some filters obtain false positives in nearly 2% of the tests run on them. These filters are used commercially, but they show two key issues for which there is no solution at present: 1) Filters are tested on standard sets of examples that are specifically designed for evaluating filters. Since the features of real-life spam are always changing, these sets do not reflect the real world where filters have to operate with any degree of certainty, and 2) In response to the acceptable performance of some filters, spammers have hit upon methods of circumvention. They study the techniques filters use, and then create masses of “suicide” e-mail (messages intended to be filtered out), so that the filters will learn from them. Next, the spammers

generate new messages that are completely different in terms of content and format. This is the major spam battlefield.

This paper describes a client-side system called JUNKER, which was designed and built to detect and filter unsolicited e-mail automatically, using several sources of knowledge that are handled by a single processing method. The system obtains optimum results and is highly effective at classifying e-mail, and highly efficient at managing resources. It is a dynamic interactive system that learns from and evolves with the evolution of spam. The user owns the control over the e-mails that he/she receives by making the decision about which e-mails he/she wants to receive. The system can learn directly from data in a user's mail inbox and this system can be customized to the user's particular preferences.

2 The JUNKER System

JUNKER is based on a hybrid filtering method that employs a novel way of filtering based on content and origin. JUNKER architecture is composed of a heuristic knowledge base and a Bayesian filter. JUNKER classifies the three parts of e-mails in the same way and then integrates the classifications resulting from each part before making a final decision about the class of e-mails.

Usually, when a Bayesian filter is trained using a historic body of valid and invalid e-mail messages, a vocabulary of valid and invalid words is created. JUNKER does not require a training stage designed to create an exhaustive vocabulary that is obsolete within a short period of time. JUNKER learns the vocabulary incrementally starting from a previously extracted knowledge base, which is formed by a set of rules and set of heuristic words without associated semantics. This word set includes words that are invalid because their morphology does not meet the morphological rules of all the languages belonging to the Indo-European family. The e-mails containing these types of invalid words are primarily conceived to fool spam filters.

2.1 Heuristic Knowledge Base

Different algorithms exist for automatically acquiring grammars. Dupont [8] proposes a general scheme for selecting the most appropriate algorithm that infers a grammar with a certain representation under different conditions. According to these ideas, the Error Correcting Grammatical Inference (ECGI) algorithm was selected for inferring a grammar that JUNKER uses to recognize well-formed words. ECGI [17] focuses on a heuristic construction of a regular grammar so that the resulting automata representing the grammar allows general and flexible recognition.

The finite state automata is used to automatically identify the words or tokens that are formed correctly and to differentiate them from invalid words. A well-formed word is composed of a term sequence. A term can be a consonant ("c"), a vowel ("v"), a number ("n"), or a symbol ("s").

The automata is created from a set of examples of well-formed words collected randomly from a set of dictionaries of several Indo-European languages. For example, the valid word "scientific-technical", represented as the string of terms "c c v v c c v c v c s c v c c c v c v c", should be recognized by the automata. If the automata

recognizes a word, then it is a well-formed word. Words taken from e-mails labelled as spam, like "v1@gra" represented by the string "c n s c c v", are not recognized by the automata as valid words, and are thus identified as misleading words.

The strings of terms that are not recognized as valid words are represented according to different parameters or criteria, including length, the type of terms contained, or the adjacency of the terms, among others. An unsupervised learning algorithm is used [13] to build a set of clusters and their descriptions. Every cluster and its description is represented by a rule relating one or more morphological criteria with a label or heuristic word. Thus, the content of the heuristic knowledge base is a set of heuristic rules whose left-hand sides evaluate some morphological criteria in words and whose right-hand sides are heuristic words:

$$Rule_i\text{: } ((Morphological\ Criterion)_i, (Heuristic\ Word)_i)$$

An example of two possible rules of this base may be written as:

Rule 1: ((number of consonants running together in a word is higher than 4), (Non-sense word 1))

Rule 2: ((number of accents in a word is higher than 3), (Non-sense word 2))

The heuristic words, i.e., Non-sense words, constitute the initial vocabulary of the Bayesian filter and it is the same for all the system end-users.

2.2 Bayesian Filtering

The filter was developed to identify and filter e-mail based on the Naïve Bayes statistical classification model [14]. This method can adapt to and learn from errors, and performs well when dealing with high-dimension data.

In general, a Bayesian classifier learns to predict the category of a text from a set of training texts that are labelled with actual categories. In the training stage, the probabilities for each word conditioned to each thematic category are estimated, and a vocabulary of words with their associated probabilities is created. The filter classifies a new text into a category by estimating the probability of the text for each possible category C_j, defined as $P(C_j \mid text) = P(C_j) \cdot \Pi_i P(word_i \mid C_j)$, where $word_i$ represents each word contained in the text to be classified. Once these computations have been carried out, the Bayesian classifier assigns the text to the category that has the highest probability value. The effectiveness of the classifier, measured by *precision* (percentage of predicted documents for a category that are correctly classified) and *recall* (percentage of documents for a category that are correctly classified), is calculated on a test set of documents with known thematic categories.

The vocabulary required by JUNKER to begin to classify e-mails is formed by the heuristic words. Initially, every heuristic word has spam and non-spam probabilities fixed beforehand. The initial value for the spam probability (*Psp*) of heuristic words is greater than the initial value for their non-spam probability (*Pnsp*). When JUNKER analyses the words of a text to be classified, it checks whether a word matches the left-hand side of any rule. If this is the case, the system substitutes the word for a heuristic word. When a word fulfils more than one rule, the system assigns the heuristic word with the lowest spam probability value to the invalid word. This bias aims to generate the minimum number of false positives. Next, the Bayesian filter uses the probabilities of the heuristic word in the same way as the valid words present

in both the text and vocabulary in order to classify the text. For example, when JUNKER receives the following text to classify: {*youuuuu, play, game*} and its vocabulary content is {(*Non-sense w1*, (Psp-w1, Pnsp-w1)), (*Non-sense w2*, (Psp-w2, Pnsp-w2)), (*play*, (Psp-play, Pnsp-play)), (*piano*, (Psp-piano, Pnsp-piano))}, it finds that "*youuuuu*" matches the heuristic word Non-sense w1 and "*play*" belongs to both the text and the vocabulary. Next, JUNKER computes the spam and non-spam probabilities of the text as P (SP | text) = P (SP) . Psp-w1 . Psp-play and P(NSP | text) = P (NSP) . Pnsp-w1 . Pnsp-play.

In order for the filter to adapt to e-mail evolution and thus maintain its performance level, the filter must evolve. The user interacts with the system by prompting false positives and negatives so that the system learns incrementally from them, either after classification has just been done or periodically. The heuristic vocabulary is just the initial state of the system vocabulary when the Bayesian filter begins to operate. As the classifier system learns, the vocabulary is updated, in terms of the number of words and word-probability values for both types of words, heuristic and learned words, and the system learns based on user prompts after classifying with an interactive interface.

2.3 Integrated Content Classification

An e-mail message can be seen as a text document composed of three separate parts: the sender, the subject, and the body of the message. Most content-based e-mail classifier systems analyse all of the parts as a single vector of words.

The design and development of the JUNKER classifier system is based on the assumption that in most cases a user can detect unsolicited e-mail just by looking at the sender and subject parts of the message. Accordingly, the system has been conceived to analyse and classify each part of the message separately. The final category of the message is the weighted integration of the resulting classifications for each part.

Since Bayesian filters are known to yield successful results, the classifier system applies a Bayesian filter to each part of the message. Each part of the message has its own vocabulary, which is initially the same as the heuristic vocabulary for the subject and body parts and is empty for the sender part. As the system learns and evolves, the various vocabularies are updated in terms of the number of words and word probabilities associated with the spam and non-spam categories.

When a new message is received, the system composes a word vector associated with each part of the message. Next, the filter computes the *Psp* and *Pnsp* probabilities for every vector by consulting the corresponding vocabulary. Any words in the message that are included in a vocabulary take on the probabilities assigned within the vocabulary. The remaining words are not considered, because they do not provide any useful information about the e-mail category.

In order to generate the minimum number of false positives, once the *Psp* and *Pnsp* probabilities have been computed for each part of the message, the system evaluates the distance between these two probability values and labels a message part as spam whenever this distance is greater than an empirically determined threshold, as follows in equation:

$$\text{Distance}_i\,(\text{Psp}\,(\text{part}_i) \mid \text{Pnsp}\,(\text{part}_i)) > u_1 \Rightarrow \text{Category}\,(\text{part}_i) = \text{spam} \qquad (1)$$

Thus, the system creates a bias in order to avoid generating false positives. After the system has analysed and computed the distance between the spam and non-spam categories for all three parts of the message, it computes the final category of the message by weighting the distances of all of the parts, defined as:

$$\text{Distance (Psp (email)} \mid \text{Pnsp (email))} = (\sum_i w_i * \text{Distance}_i) / 3 \quad (2)$$

The sender and the subject of a message may provide the user with the most obvious clues as to the intention behind the message. This factor is taken into account in the final overall distance, because w_1 and w_2 take higher values than w_3 by default.

Since the bias against generating false positives is included at all of the system decision points, the system only classifies a message as spam when the overall distance is above a global threshold termed "filter confidence". The user can interactively modify the filter confidence.

3 Intelligent Management of Vocabularies and Resources

Once the system has classified incoming messages and the user has been informed of the resulting classification, the user can note the system errors, in real time or periodically, using a friendly interface. The interface also allows the user to remove correctly classified e-mail from the filter domain.

The properties of the system allow carrying out an intelligent vocabulary management to prevent an exhaustive increase in vocabulary. On the one hand, vocabulary upgrades do not include the new words contained in correctly classified and removed e-mail from the filter domain. The reason why such e-mail is correctly classified is that the words that are present in messages and vocabularies alike are enough to categorize the e-mail into its target class. Although increasing the vocabulary size may provide a filter with a greater capacity to discriminate, very large vocabularies require more classification time and are accordingly less efficient.

On the other hand, when the system has to learn from misclassified e-mails, the invalid words in these e-mails, which match some rule of the knowledge base and are identified as heuristic words, are not added to the vocabulary. Instead, the system updates the spam and non-spam probabilities of the heuristic words in the vocabulary that has been found in the e-mail.

The system hybrid behaviour based on filtering origin and content lies in applying the Bayesian filter to the sender part of the message first of all. If the filter finds the sender in its vocabulary, the message is directly classified as non-spam, and the system stops filtering the remaining two parts of the message. If the sender is not found, the system goes on to analyse the content of the subject and body parts of the message. The method used to integrate the classifications of all three parts by giving priority to the sender classification prevents the system from wasting processing resources on classifying e-mail and thereby increases its efficiency.

3.1 Updating the Sender, Subject, and Body Vocabularies

The Bayesian filter begins with an empty vocabulary in the sender part. The system initially classifies this part of the incoming messages into the non-spam category. The integrated classification of the three message parts is what finally categorizes e-mail

as spam or non-spam. When the user accepts the classification made by the system, the system stores only the address of the senders of non-spam e-mail whose non-spam probability is greater than its spam probability in the sender vocabulary.

When the filter has to learn from the misclassifications pointed out to it by the user, the system stores only the senders of false positives, i.e., the senders of e-mails classified as spam that are actually non-spam. The senders of false negatives, i.e., the senders of spam that is erroneously assigned to the non-spam category, are ignored, because the majority of unsolicited e-mail hardly ever comes from the same senders twice. Thus, as the system operates over time, the system builds the vocabulary of the sender part using the list of the trusted senders, or white list, which is processed by the Bayesian filter the same as the other two message parts. The initial subject and body vocabularies are formed by the heuristic words, and the system begins to classify these parts of the messages by searching for the words in the vocabularies. When the system has to learn from misclassifications highlighted by the user, these vocabularies are upgraded with the words from the misclassified messages, including false positives and negatives. These new words receive the values of the spam and non-spam probabilities that the system sets for them by default.

Both the correctly classified messages and the new words from misclassified messages prompt the filter to update the probabilities for the entire vocabularies of both parts, so that the vocabularies contain the system's current knowledge.

4 Empirical Evaluation

JUNKER was evaluated on two different set of messages. The first one, LingSpam [1], is composed of 2,412 legitimate messages and 481 spam messages received by LingSpam authors during a given period of time. The messages in LingSpam collection were pre-processed by removing the "from" part, applying a stop list, stemming the words, and removing all the invalid words from a morphological viewpoint. The corpus was split by LingSpam authors into 10 folds in order to apply a 10-fold cross validation. The second corpus, SpamAssassin Corpus [19], is composed of 4,149 legitimate messages and 1,896 spam messages that were collected from individual e-mail boxes.

The SpamAssassin corpus was not pre-processed like LingSpam although all the e-mails came from different senders. In spite of the fact that both corpora were collected from real users, they do not represent the current, actual situation of the users' mail inboxes, since these e-mails were somehow pre-processed and nearly all the noise had been removed. Thus, JUNKER is not able to test some of its most novel properties on these e-mail collections. Anyway, the system presented in this paper obtained good results on both corpora, as shown in Fig. 1 a), b), c) and d).

In [3] several techniques, ranging from Naïve Bayes to Support Vector Machines and Genetic Programming, were evaluated on the LingSpam corpus and the results obtained were nearly 99.5% recall and nearly 81% precision when classifying e-mails in the spam class. The same paper showed that if the corpus is not stemmed and stop listed, the precision in the spam class improves. The classification performance of a modified and fine-tuned Naïve Bayes algorithm evaluated on the SpamAssassin corpus was nearly 99.9% recall and 95% precision in the spam class [24].

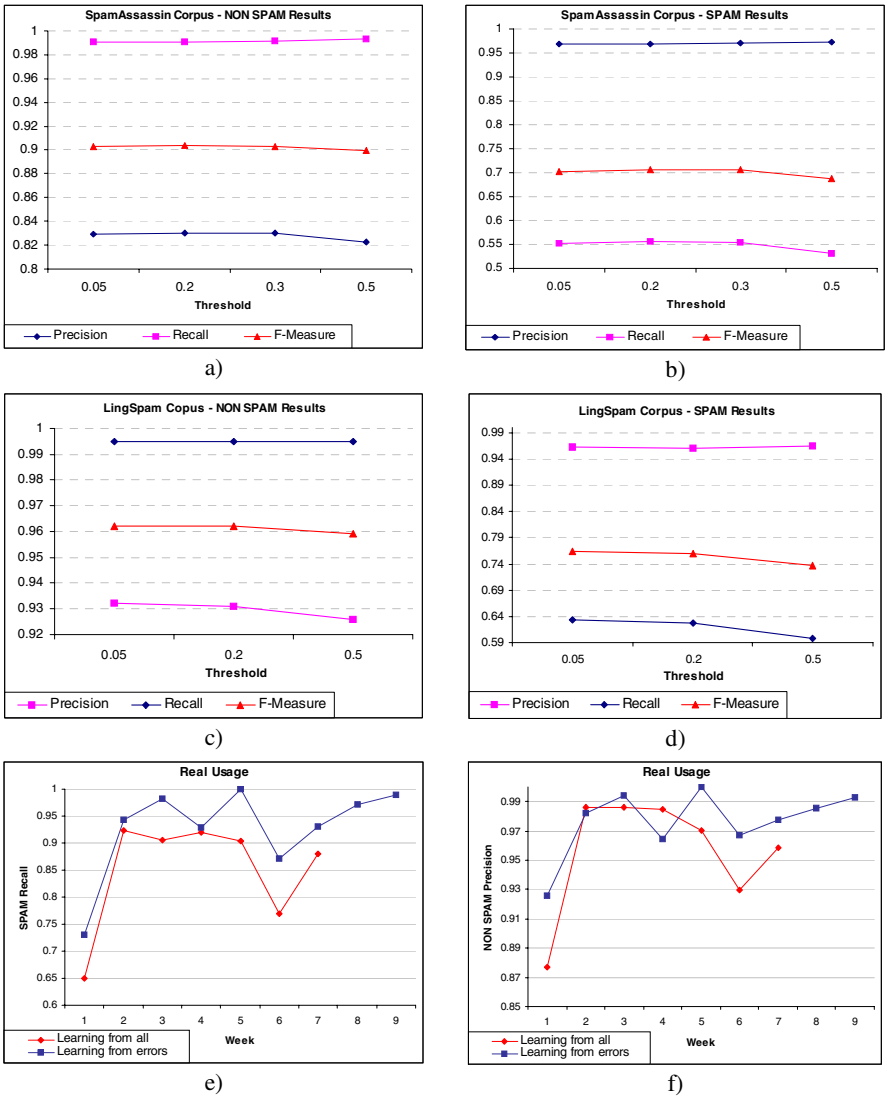

Fig. 1. JUNKER results: a) Non-Spam category on SpamAssassin Corpus, b) Spam category on SpamAssassin Corpus, c) Non-Spam category on LingSpam Corpus, d) Spam category on LingSpam Corpus, for different threshold values, and e) Spam recall and f) Non-Spam precision on real usage during a period of time

JUNKER has been also checked on real usage, by dealing with the e-mails received by the authors of this paper at real-time during a period of time. The initial heuristic vocabulary of the system consisted of 5 heuristic words. The evolution of JUNKER using a distance threshold of 0.3 has been evaluated in two ways: 1) the system only learned from misclassifications for 9 weeks, and 2) the system learned

from both correctly and misclassified e-mails for the next 7 weeks (see results in Fig. 1 e) and f)). The relation between the increase of the number of received e-mails and the size of the vocabularies is logarithmic-like. The increase of the vocabulary is smoother when the system only learns from misclassifications.

5 Conclusion

JUNKER works as a customized filter by analysing the e-mail messages of every user individually. It is an effective system, not only for avoiding the creation of false positives, but also for filtering. Its main advantage is that it slows down spammer attempts to fool the filter. Its good performance is reached without a training stage that uses e-mail that has first been received by the user. This classification performance is easier to achieve because of integrating the classifications of the three parts of each message and because of the homogeneous processing of these parts by a single Bayesian filter. The system procedure for evaluating the sender part first allows the system to give high performance in terms of resource management and in terms of response time for classifying and learning from errors.

The system features an easy, friendly interface that provides the user with a way of highlighting misclassifications and guiding the system evolution, based on the e-mail the user receives. JUNKER has been designed for client-side operation. However, thanks to its underlying inner nature, it does allow for straightforward expansion to multiple-user support.

References

1. Androutsopoulos, I., Paliouras, G., Karkaletsis, G., Sakkis, G., Spyropoulos, C., Stamatopoulos, P.: Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (2000)
2. Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G., Spyropoulos, C. D.: An Evaluation of Naive Bayesian Anti-Spam Filtering. Proc. of the workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML) (2000) 9-17
3. Carreras X., Márquez L.: Boosting Trees for Anti-Spam Email Filtering. In: Mitkov, R., Angelova, G., Bontcheva, K., Nicolov, N., Nikolov, N. (eds.). Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing. Tzigov Chark, BG (2001) 58-64
4. Cohen, W.: Learning rules that classify e-mail. AAAI Spring Symposium on Machine Learning in Information Access (1996)
5. Commission of the European Communities: Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee of the Regions on unsolicited commercial communications or 'spam', Brussels (2004)
6. Cunningham, P., Nowlan, N., Delany, S.J., Haahr M.: A Case-Based Approach to Spam Filtering that Can Track Concept Drift. Technical Report at Trinity College, TCD-CS-2003-16, Dublin (2003)

7. Daelemans, W., Zavrel, J., van der Sloot, K., van den Bosch, A.: TiMBL: Tilburg Memory-Based Learner - version 4.0 Reference Guide (2001)
8. Dupont, P.: Inductive and Statistical Learning of Formal Grammars. Technical Report, research talk, Department of Ingenerie Informatique, Universite Catholique de Louvain (2002)
9. Drucker, H., Wu, D., Vapnik, V. N.: Support Vector Machines for Spam Categorization, IEEE Transactions on Neural Networks, 10(5) (1999)
10. Graham, P.: A plan for spam. (2002), http://www.paulgraham.com/spam.html
11. Graham, P.: Better Bayesian Filtering. Proc. of Spam Conference 2003, MIT Media Lab., Cambridge (2003)
12. Mertz, D.: Spam Filtering Techniques. Six approaches to eliminating unwanted e-mail. Gnosis Software Inc. (2002)
13. Michalsky R.S.: A theory and methodology of inductive learning. In: Michalsky R.S., Carbonell J.G., and Mitchell T.M. (eds.): Machine Learning: An Artificial Intelligence Approach. Springer-Verlag (1983) 83-134
14. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
15. Pyzor, http://pyzor.sourceforge.net
16. Randazzese, V. A.: ChoiceMail Eases Antispam Software Use While Effectively Figthing Off Unwanted E-mail Traffic. CRN (2004)
17. Rulot, H.: ECGI. Un algoritmo de Inferencia Gramatical mediante Corrección de Errores. Phd Thesis, Facultad de Ciencias Físicas, Universidad de Valencia (1992)
18. Sergeant, M.: Internet-Level Spam Detection and SpamAssassin 2.50. Proceedings of Spam Conference 2003, MIT Media Lab. Cambridge (2003) http://spamassassin.org
19. http://www.spamassassin.apache.org
20. Tagged Message Delivery Agent Homepage, http://tmda.net
21. Teredesai, A., Dawara, S.: Junk Mail, a Bane to Messaging. Technical Report of STARE Project, Rochester Institute of Technology, http://www.cs.rit.edu/~sgd9494/STARE.htm, (2003)
22. Vinther, M.: Junk Detection using neural networks. MeeSoft Technical Report (2002) http://logicnet.dk/reports/JunkDetection/JunkDetection.htm
23. Vipul's Razor, http://razor.sourceforge.net
24. Yerazunis, W. S.: The Spam-Filtering Accuracy Plateau at 99,9% Accuracy and How to Get Past It. Proceedings of MIT Spam Conference (2004)