Analyzing the Impact of Corpus Preprocessing on Anti-Spam Filtering Software

J. R. Méndez[1], E. L. Iglesias[1], F. Fdez-Riverola[1], F. Díaz[2], J.M. Corchado[3]

[1] Dept. Informática, University of Vigo, Escuela Superior de Ingeniería Informática, Edificio Politécnico, Campus Universitario As Lagoas s/n, 32004, Ourense, Spain
{moncho.mendez | eva | riverola}@uvigo.es
[2] Dept. Informática, University of Valladolid, Escuela Universitaria de Informática, Plaza Santa Eulalia, 9-11, 40005, Segovia, Spain
fdiaz@infor.uva.es
[3] Dept. Informática y Automática, University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain
corchado@usal.es

Abstract. Because of the volume of spam e-mail and its evolving nature, many statistical techniques have been applied until now for the construction of anti-spam filtering software. In order to train and test filters, it is necessary to have a large e-mail corpus. In this paper we discuss several considerations that researchers must take into account when building and processing a corpus. After reviewing several text preprocessing methods used on spam filtering, we show the results obtained by different machine and lazy learning approaches when the preprocessing of the training corpus changes. The results obtained from the experiments carried out are very informative and they back up the idea that instance-based reasoning systems can offer significant advantages in the spam filtering domain.

1 Introduction and Motivation

Unsolicited e-mail messages, also known as spam, have become a serious problem for internet users. Recent studies show that between one-seventh and one-half of e-mail messages going into internet user inboxes are spam [1].

Nowadays anti-spam filtering software seems to be the most viable solution to the spam problem. It works attempting to automatically identify an incoming e-mail message using different approaches [2] and classifying it as either 'spam' or 'legitimate'. Spam filtering methods are often classified as rule-based or content-based (statistical). The first ones classify documents based on whether or not they meet a particular set of criteria [3]. The last ones do not require specifying any rules explicitly. They are primary driven by statistics that can be derived from the content of the messages (i.e., word frequency) [4].

In our work we identify two main types of content-based techniques: (*i*) machine learning (ML) algorithms and (*ii*) memory-based and case-based reasoning approaches. ML approaches use an algorithm to 'learn' the classification from a set of training messages. On the other hand, memory-based and case-based reasoning techniques store all training instances in a memory structure and try to classify new messages finding similar e-mails on it. Hence, the decision of how to solve a problem is deferred until the last moment.

Content-based filters tend to be more successful than rules-based ones. It is more difficult for spammers to create a spam e-mail because they have no idea what type of e-mail each user is training their filters on. Furthermore, the lazy nature of memory-based and case-based approaches makes them specially suitable for dynamic environments as spam [5, 6].

In order to train and test content-based filters, it is necessary to build a large corpus with spam and legitimate e-mails or use a public corpus. Anyway, e-mails have to be preprocessed to extract their words (*features*). Also, since the number of features in a corpus can end up being very high, it will generally be necessary to choose those features that better represent e-mails before carrying out the filter training to prevent the classifiers from overfitting.

The effectiveness of content-based anti-spam filters relies on the appropriate choice of the features. If the features are chosen so that they may exist both in a spam and legitimate messages then, no matter how good learning algorithm is, it will make mistakes. Therefore, the preprocessing step of e-mail features extraction and the later selection of the most representative are crucial for the performance of the filter.

Our main goal in this paper is the evaluation and comparison of different feature extraction techniques used in text categorization. We will analyze what are their strengths and weaknesses when they are applied to the spam problem domain. Therefore, we will show the results obtained by different well-known content-based techniques when the preprocessing of the training corpus changes. The selected models go from the utilization of Naïve Bayes [7], boosting trees [8], Support Vector Machines [9] to three case-based systems for spam filtering that can learn dynamically: a Cunningham *et al.* system which we call *Cunn Odds Rate* [10], its improved version named ECUE [11] and the SpamHunting system [12].

The rest of the paper is organized as follows: Section 2 introduces previous work on machine learning and case-based e-mail filters. Section 3 describes the selected public available corpus for empirical model evaluation. Section 4 discuses several issues related with message representation and feature selection. Section 5 presents the experiments carried out and the results obtained and discusses the major findings. Finally, Section 6 exposes the main conclusions reached by the analysis of the experiments carried out.

2 Spam Filtering Techniques

2.1 Machine Learning Approaches

The most popular classical filtering models are bayesian methods. Bayesian filtering is based on the principle that most of the events are conditioned. So, the probability that an event happens can be deduced from the previous appearances of that event. This technique can be used to classify spam. If some feature is often in spam but not in legitimate e-mails, then it would be reasonable to assume that an e-mail including this feature will be probably spam. Although there are several approaches of the bayesian method, the most widely used to spam filtering is Naïve Bayes algorithm [7]. Besides bayesian models, Support Vector Machines (SVM) and boosting techniques are also well-known ML techniques used in this field.

SVMs [9] are based on representing e-mails as points in an n-dimensional space and finding an hyperplane that generates the largest margin between the data points in the positive class and those in the negative class. SVM has become very popular in the ML and DM community due to its good generalization performance and its ability to handle high-dimensional data through the use of kernels. Some implementations of SVM can be found in ML environments such as Waikato Environment for Knowledge Analysis[1] (WEKA) or Yet Another Learning Environment[2] (YALE). Particularly, WEKA includes the *Sequential Minimal Optimization* (SMO) algorithm which has demonstrated a good trade-off between accuracy and speed (see [13] for details).

Boosting algorithms [8] are techniques based on the use of weak learners; that is to say, algorithms that learn with a next error rate to 50%. The main idea of boosting is to combine the hypotheses to one final hypothesis, in order to achieve higher accuracy than the weak learner's hypothesis would have. Different boosting algorithms have been developed for classification tasks, so much binary as multi-class. Among them we could highlight Adaboost [14].

Recently, several new ML models has been introduced for e-mail classification such as Chung-Kwei [15], which is based on pattern-discovery. As well as it is faster than other ML approaches, it becomes better on performance.

2.2 Case-based Reasoning Approaches

Case-based approaches outperform previous techniques in anti-spam filtering [11]. Case-based classification works well for disjoint concepts as spam (spam about *porn* has little in common with spam offering *rolex*) whereas ML techniques try to learn a unified concept description. Another important advantage of this approach is the ease with which it can be updated to tackle the *concept drift* problem in the anti-spam domain [6].

Delany *et al*. present in [11] a case-based system for anti-spam filtering called ECUE (*E-mail Classification Using Examples*) that can learn dynamically. The system use a similarity retrieval algorithm based on Case Retrieval Net*s* (CRN) [16]. CRN networks are equivalent to the k-nearest neighbourhood algorithm but are computationally more efficient in domains where there is feature-value redundancy and missing features in cases, as spam. ECUE classifier use unanimous voting to determine whether a new e-mail is spam or not. All the returned neighbours need to be classified as spam e-mails in order to classify as spam the new message. The ECUE

[1] WEKA is available from http://www.cs.waikato.ac.nz/ml/weka/

[2] YALE is available from http://yale.sourceforge.net

system represents the evolution from *Cunn Odds Rate* [10], a previous successful system of the same authors.

Also, in [12] a lazy learning hybrid system is introduced to accurately solve the problem of spam labelling and filtering. The model, named SpamHunting, follows an Instance-Based Reasoning (IBR) approach. According to this, SpamHunting uses an instance memory structure as primary way of manage knowledge. The retrieval stage is carried out using a novel dynamic *k*-NN Enhanced Instance Retrieval Network (EIRN). The EIRN network facilitates the indexation of instances and the selection of those that are most similar to the new e-mail. Similarity between two given e-mails is measured by the number of relevant features found in both messages. EIRN can quickly retrieve all stored e-mails having at least one shared feature with a target message. The reuse of similar messages is done by means of a simple unanimous voting mechanism to determine whether the target case is spam or not. The revision stage is only carried out in the case of unclassified messages, where the system employs general knowledge in the form of meta-rules extracted from the e-mail headers to assign a final class.

3 Benchmark Corpus for Spam Filtering Research

As previously mentioned, it is essential to provide content-based filters with an appropriate corpus of e-mails for training and testing purposes. The corpus should be made up of both spam and legitimate e-mails. Each message should be marked as being either spam or non-spam. By training the filters on this corpus, they should learn the main characteristics that differentiate spam from legitimate messages.

Table 1. Temporal distribution of messages belonging to the SpamAssassin corpora of emails

	Jan 02	Feb 02	Mar 02	Apr 02	May 02	Jun 02	Jul 02	Aug 02	Sep 02	Oct 02	Nov 02	Dec 02	Sum 02
Spam	2	1	0	0	0	1	8	182	276	5	0	0	475
Error	-	-	-	-	-	-	-	-	-	-	-	-	26
Legitim.	0	44	0	0	2	5	157	561	1272	757	0	0	2798
Total													3299
	Jan 03	Feb 03	Mar 03	Apr 03	May 03	Jun 03	Jul 03	Aug 03	Sep 03	Oct 03	Nov 03	Dec 03	Sum 03
Spam	0	12	18	18	312	145	496	330	274	6	9	25	1645
Error	-	-	-	-	-	-	-	-	-	-	-	-	239
Legitim.	1	44	0	0	2	7	704	1334	1239	727	36	55	4149
Total													6033

Despite privacy issues related with the content of a message, there are several public available corpora of e-mails just as LingSpam[3], PU[3], JunkeE-mail[4], DivMod[5] or

[3] Available at http://www.iit.demokritos.gr/

[4] Available at http://clg.wlv.ac.uk/projects/junk-e-mail/

SpamAssassin[6]. In our work, we use the SpamAssassin corpora. It contains 9332 different messages from January 2002 up to and including December 2003 distributed as Table 1 shows. The error row shows the presence of messages with a corrupt date.

4 Message Representation

In order to increase message manipulation speed as well as knowledge representation ability, messages should not be stored in its primitive form, as they are in a original corpus. It is necessary to convert each message into a reliable message descriptor which can be easily assembled. In learning algorithms, training messages are usually represented as a vector $\vec{t} = \left\langle t_1, t_2, ..., t_p \right\rangle$ of weighted terms, T_i, much as in the vector space model in information retrieval [17, 18].

Features can be identified using a variety of generic lexical tools, primarily by tokenising the e-mail into words. At first glance, all that seems to be involved in it is the recognition of spaces as word separators. However, at least the following particular cases have to be considered with care: hyphens, punctuation marks, and the case of the letters (lower and upper case).

Lexical analyzer normally breaks hyphenated words and remove punctuation marks. However, in the spam domain many of these symbols are among the best discriminating attributes in a corpora, because they are more common in spam messages than legitimate ones. For this reason, hyphens and punctuation marks are not removed here. On the subject of case, the lexical analyzer normally converts all the text to either lower or upper case. It is also done here to reduce the number of terms.

In text categorization it is common to reduce the set of representative terms with very large collections [18]. This can be accomplished through the elimination of *stopword* (such as articles and connectives) and the use of *stemming* (which reduces distinct words to their common grammatical root). Since spam is a special form of text categorization, it could be applied here also.

Once carried out the lexical analysis over the corpus, the weight of terms in each message *e*, need to be calculated. The measure of the weight can be (*i*) binary (1 if the term occurs in the message, 0 otherwise), (*ii*) the *term frequency* (TF) representing the number of times the term occurs in the message calculated by Expression (1) or (*iii*) the *inverse document frequency* (IDF) given by Expression (2) denoting those terms that are common across the messages of the training collection.

$$t_i(e) = \frac{n_i(e)}{N(e)} \tag{1}$$

$$t_i(e) = \frac{n_i(e)}{N(e)} \log_2 \frac{m}{df(T_i)} \tag{2}$$

[5] Available at http://www.divmod.org/cvs/corpus/spam/

[6] Available at http://www.spamassassin.org/publiccorpus/

In Equations (1) and (2), $n_i(e)$ is the number of occurrences of term T_i in e, $N(e)$ represents the total number of occurrences of terms in e, m is the number of training messages and $df(T_i)$ stands for the number of training messages where the term T_i occurs.

5 Performance Evaluation

The final goal of our experiments is to measure the impact of applying different preprocessing steps over the corpus before training the models. The experiments have been done using Naïve Bayes, a SMO implementation of SVM, Adaboost, and the three previously commented CBR systems: ECUE, *Cunn Odds Rate* and SpamHunting.

Six well-known metrics [4] have been used in order to evaluate the performance (efficacy) of all the analyzed models: percentage of correct classifications (%OK), percentage of False Positives (%FP), percentage of False Negatives (%FN), spam *recall*, spam *precision* and *total cost ratio* (TCR) with three different cost values. The experiments were carried out at three different preprocessing scenarios: (*i*) applying stopword removal and stemming analysis, (*ii*) applying stopword but without stemming and (*iii*) without applying neither stopword nor stemming.

Typically, message representation scheme presented in Section 4 leads to very high-dimensional feature spaces. Several authors have noted the need for feature selection in order to make possible the use of conventional ML techniques, to improve generalization accuracy and to avoid over fitting of the models. Following the recommendation of [19], the *information gain* (IG) criterion is usually used to select a representative subset of features.

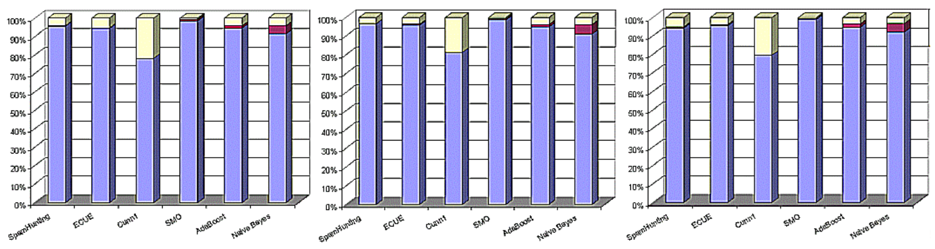
All the analyzed models except from *Cunn Odds Rate* and SpamHunting systems use IG to select the most predictive features as it has been shown to be an effective technique in aggressive feature removal in text classification [19]. For our comparisons, we have selected the best performance model of each technique varying between 100 and 2000 features. For *Cunn Odds Rate* model, we have maintained the original technique of selecting 30 words for representing spam e-mails plus 30 words representing legitimate messages. The algorithm employed for sorting the vocabulary is based on the odds-ratio described in [11].

SpamHunting terms selection is not made using the vocabulary of the whole corpus. Instead of this, each message has its own relevant terms. The relevant feature list of each message is computed as the minimum set containing the most frequent features of the specified e-mail, which frequency amount is greater than a specified threshold in the range [0,1]. As the best results have been obtained using the 30% frequency amount, we computed the relevant feature list as the most repeated features whose frequency amount is greater than mentioned threshold.

All the experiments have been carried out using a 10-fold stratified cross-validation [20] in order to increase the confidence level of results obtained.

5.1 Experimental Results

Figure 1 shows the percentage of correct classifications, false positive rate and false negative rate belonging to the six analyzed models over the defined scenarios. Analyzing Figure 1 we can realize that Naïve Bayes and SVM techniques get better performance with no stemming and no stopword removal (Scenario 3). When stopword and stemming are used (Scenario 1) SpamHunting, ECUE and Adaboost report better accuracy but generally worst results. However, applying only stopword removal (Scenario 2) in those models that do not incorporate IG for feature selection leads to a significant accuracy increment. In order to facilitate a deeper analysis, Table 2 shows the mean values over the 10 fold-cross validation for the scores presented in Figure 1.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 1. Comparative model performance varying preprocessing steps

Table 2. Mean value of correct clasifications, FPs and FNs with 10 fold-cross validation

		Naïve Bayes	**Ada boost**	**SVM**	**Cunn Odds Rate**	**ECUE**	**Spam Hunting**
Scenario 1	OK	852.5	881.9	920.2	730.4	880.2	892.9
	False Positives	43	12.2	7.8	0.5	3.6	1.5
	False Negatives	37.7	39.1	5.2	202.3	49.4	38.8
Scenario 2	OK	849.6	885.1	919.1	758.3	893	900.8
	False Positives	48.6	13.4	8.7	0.2	6.1	1.8
	False Negatives	35	34.7	5.4	174.7	34.1	30.6
Scenario 3	OK	857.9	883.4	922.2	742.1	889.8	880.3
	False Positives	44	16	5.3	0.1	6.8	4.2
	False Negatives	31.3	33.8	5.7	191	36.6	48.7

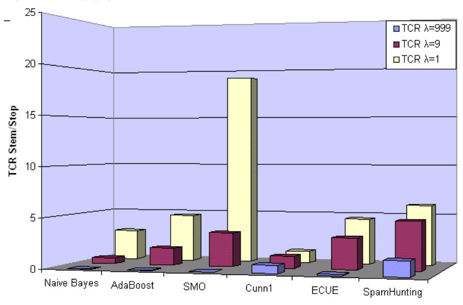
Table 3 shows a comparative study between the three proposed scenarios using recall and precision scores. Results on recall evaluation using classical ML models are better when no stopword removal and no stemming is used (Scenario 3). However, CBR/IBR approaches become better when only stopword removal is performed (Scenario 2) while the worst results are obtained when both stemming and stopword removal is applied (Scenario 1).

Table 3. Averaged precision and recall scores over 10 fold-cross validation

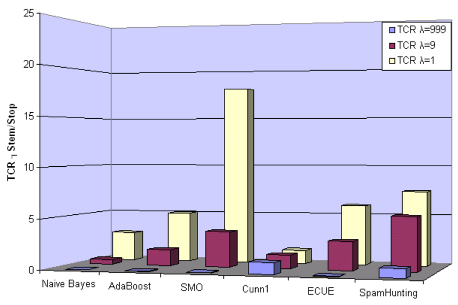
		Naïve Bayes	**Ada boost**	**SVM**	**Cunn Odds Rate**	**ECUE**	**Spam Hunting**
Scenario 1	Recall	0.842	0.836	0.978	0.150	0.793	0.837
	Precision	0.824	0.943	0.968	0.986	0.981	0.993
Scenario 2	Recall	0.853	0.854	0.977	0.266	0.857	0.871
	Precision	0.807	0.939	0.964	0.997	0.971	0.991
Scenario 3	Recall	0.869	0.858	0.976	0.198	0.846	0.795
	Precision	0.824	0.928	0.978	0.998	0.967	0.978

Analyzing in Table 3 precision scores gathered from experiments, we can realize that it is possible to obtain better results when stopword removal and stemming is applied (Scenario 1) except for SVM and *Cunn Odds Rate* models. These techniques work better without any preprocessing step (Scenario 3).

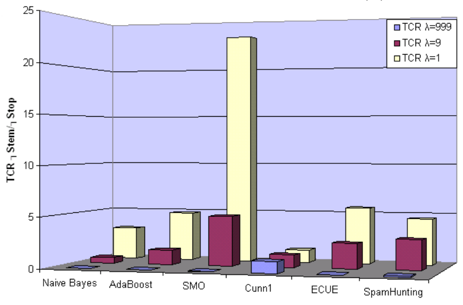
In order to compare the performance of the models taking into account the three predefined scenarios but with a cost-sensitive point of view, we calculate the TCR score in three different situations. TCR assumes than FP errors are λ times more costly than FN errors, where λ depends on the usage scenario (see [4] for more details). In the experiments carried out in this paper, the values for λ parameter were 1, 9 and 999.



(a) TCR in Scenario 1



(b) TCR in Scenario 2



(c) TCR in Scenario 3

Fig. 2. TCR score graphics varying λ (equal to 1, 9, 999) value in the three different scenarios

Figure 2 shows the results taking into account the TCR score and varying the λ parameter. The best performance on classical ML approaches are obtained when no stemming and no stopword removal are applied (Scenario 3) except for Adaboost, that increases its TCR score when only stopword are applied (Scenario 2). By other side, CBR/IBR approaches work better if stopword are applied to the corpus (Scenarios 1 and 2).

6 Conclusions

In this paper we have analyzed the impact of corpus preprocessing in the performance of anti-spam filtering software. For this task, we have briefly revised the most popular filtering techniques from the ML community as well as CBR and IBR previous successful implemented systems. Before defining the experiments to carry out, we have presented the benchmark corpora of e-mails and discussed several issues about message representation.

In order to carry out the experiments, we have considered three different scenarios and six standard scores to measure performance among the models. 10-fold stratified cross-validation was used in order to increase the confidence level of results obtained. From the analysis of these results we can infer valuable information about the preprocessing needed in order to construct accurate anti-spam filtering models.

Firstly, classical ML based models can get the best number of correctly classified messages by removing all preprocessing steps, but stopword removal and stemming is recommended if best accuracy is needed. Secondly, CBR/IBR models can obtain better performance by stopword removal although stemming can improve accuracy.

Moreover, applying stemming can significantly reduce the number of selected features belonging to the corpus. According to this, it will also decrease the time needed to compute IG for all features and the spam recall score. Nevertheless, as results show, if stemming is applied the models will obtain the smallest amount of correct classifications excluding the SpamHunting system.

If the main goal is the minimization of the FP errors among the models, the results obtained from experiments suggest that stemming should be used. This idea is backed up because legitimate messages are better classified when stemming is used (by the successful identification of semantic roots belonging to legitimate messages). This fact helps the models to better differentiate spam from legitimate e-mails. In addition, if the main goal is the improvement in correct classification rate (therefore diminishing the total number of errors), then stemming is not recommended.

It is important to highlight that depending on the model, different results are obtained when changing the preprocessing steps carried out over the whole corpus. So, these issues need to be kept in mind in both training the model comparing its accuracy with another anti-spam classifiers.

The main conclusion of this work is that the effort on stemming and removing stopwords does not pay in improvement of the current algorithms for spam detection. In addition, spam producers are very creative, and learning from a static corpus (preprocessed or not) seems to be a now a naive approach.

References

1. Vaknin, S.: The Economics of Spam. United Press International (2002)
2. Michelakis, E., Androutsopoulos, I., Paliouras, G., Sakkis, G., Stamatopoulos, P.: Filtron: A Learning-Based Anti-Spam Filter. Proc. of the First Conference on E-mail and Anti-Spam CEAS, (2004)
3. Wittel, G.L., Wu, S.F.: On Attacking Statistical Spam Filters. Proc. of the First Conference on E-mail and Anti-Spam CEAS, (2004)
4. Androutsopoulos, I., Paliouras, G., Michelakis, E.: Learning to Filter Unsolicited Commercial E-Mail. Technical Report 2004/2, NCSR "Demokritos" (2004)
5. Kelly, M.G., Hand, D.J., Adams, N.M.: The impact of changing populations on classifier performance. Proc. of the 5th International Conference on Knowledge Discovery and Data Mining, ACM Press, (1999) 367–371
6. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning, Vol. 23 (1). (1996) 69–101
7. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian approach to filtering junk e-mail. In Learning for Text Categorization – Papers from the AAAI Workshop, (1998) 55–62, Technical Report WS-98-05 AAAI
8. Carreras, X., Màrquez, L.: Boosting trees for anti-spam e-mail filtering. Proc. of the 4th International Conference on Recent Advances in Natural Language Processing, (2001) 58–64
9. Vapnik, V.: The Nature of Statistical Learning Theory. 2nd Ed. Statistics for Engineering and Information Science, Springer, New York (1999)
10. Delany, S.J., Cunningham P., Coyle L.: An Assessment of Case-base Reasoning for Spam Filtering. Proc. of Fifteenth Irish Conference on Artificial Intelligence and Cognitive Science: AICS-04, (2004) 9–18
11. Cunningham, P., Nowlan, N., Delany, S.J., Haahr, M.: A Case-Based Approach to Spam Filtering that Can Track Concept Drift. Proc. of the ICCBR'03 Workshop on Long-Lived CBR Systems, (2003)
12. Fdez-Riverola, F., Méndez, J.R, Iglesias, E.L., Díaz, F.: Representación flexible de e-mails para la construcción de filtros anti-spam: un caso práctico. Proc. of the VI Jornadas de Transferencia Tecnológica de Inteligencia Artificial. *To appear.*
13. Platt, J.: Fast training of Support Vector Machines using Sequential Minimal Optimization. In Sholkopf, B., Burges, C., Smola, A. (eds.). Advances in Kernel Methods – Support Vector Learning, MIT Press, (1999) 185–208
14. Schapire, R.E., Singer, Y.: BoosTexter: a boosting-based system for text categorization. Machine Learning, Vol. 39 (2/3). (2000) 135–168
15. Isidore Rigoutsos and Tien Huynh. Chung-Kwei: a Pattern-discovery-based System for the Automatic Identification of Unsolicited E-mail Messages (SPAM). Proc. of the First Conference on E-mail and Anti-Spam CEAS, (2004)
16. Lenz, M., Auriol, E., Manago, M.: Diagnosis and Decision Support. Case-Based Reasoning Technology. Lecture Notes in Artificial Intelligence, Vol. 1400, (1998) 51–90
17. Salton, G., McGill, M.: Introduction to modern information retrieval, McGraw-Hill, (1983)
18. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley, (1999)
19. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. Proc. of the Fourteenth International Conference on Machine Learning: ICML-97, (1997) 412–420
20. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence: IJCAI-95, (1995) 1137–1143