required in the near future. We are currently exploring the usage of simulations to learn the value of team activity (perception to action transitions), given that agents acquire too much sensory raw data and that their perceptual limitations is likely to hide environmental crucial features.

Acknowledgements

This research has been partially supported by the program PRODEP III 5.3/13/03.

References

- [1] N. Schurr, S. Okamoto, R. Maheswaran, P. Scerri, and M. Tambe, *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, chapter entitled *Evolution of a Teamwork Model*, Cambridge University Press, 2004.
- [2] A. Rao and M. Georgeff, "Modeling rational agents within a BDI-architecture," in *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR&R-91)*, Morgan Kaufmann Publishers: San Mateo, CA, pp. 473-484, April 1991.
- [3] B. Grosz and S. Kraus, "Collaborative plans for complex group action," *Artificial Intelligence*, vol. 86(2), pp. 269-357, 1996.
- [4] R. Nair, M. Tambe, and S. Marsella, "Team formation for reformation in multi-agent domains like RoboCupRescue," in G. Kaminka, P. Lima and R. Rojas (eds.), *RoboCup 2002: Robot Soccer World Cup VI. Lecture Notes in Computer Science #2752*, Springer pp. 150-161, 2003.
- [5] H. Kiano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou and S. Shimada, "RoboCupRescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research," In *Proceedings of IEEE International Conference on Man, System and Cybernetics (MSC-99)*, vol. VI, pp. 739-743, Tokyo, Japan October 12-15, 1999.
- [6] S. Tadokoro, H. Kiano, T. Takahashi, I. Noda, H. Matsubara, A. Shinjou, T. Koto, I. Takeuchi, H. Takahashi, F. Matsuno, M. Hatayama, J. Nobe and S. Shinada, "The RoboCup-Rescue Project: A Robotic Approach to the Disaster Mitigation Problem," In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA00)*, pp. 4090-4095, April 23-28, San Francisco, USA, 2000.
- [7] P. Trigo, P. Araújo, A. Remédios, C. Lopes, B. Bastião, T. Loureiro, L. Moniz and H. Coelho, "The 5Rings Team Description Paper," In *Proceedings of the RoboCup2004 Symposium, Team Description Papers*, July 4-5, Lisbon, Portugal, 2004.
- [8] P. Trigo, H. Coelho, "The Multi-Team Formation Precursor of Teamwork," In *Proceedings of the EPIA05, LNAI 3808*, Springer-Verlag, 2005.
- [9] P. Cohen and H. Levesque, "Teamwork," *Special Issue on Cognitive Science and Artificial Intelligence, No.4*, vol. 25(4), pp. 487-512, 1991.

Tokenising, Stemming and Stopword Removal on Anti-spam Filtering Domain

J. R. Méndez¹, E. L. Iglesias¹, F. Fdez-Riverola¹, F. Díaz², and J.M. Corchado³

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. Junk e-mail detection and filtering can be considered a cost-sensitive classification problem. Nevertheless, preprocessing methods and noise reduction strategies used to enhance the computational efficiency in text classification cannot be so efficient in e-mail filtering. This fact is demonstrated here where a comparative study of the use of stopword removal, stemming and different tokenising schemes is presented. The final goal is to preprocess the training e-mail corpora of several content-based techniques for spam filtering (machine approaches and case-based systems). Soundness conclusions are extracted from the experiments carried out where different scenarios are taken into consideration.

1 Introduction

Removal of non-informative words and *stemming* are document preprocessing techniques commonly used in text classification and text filtering [1, 4] to improve the accuracy of the results and to reduce the redundancy of the computation. Non-informative words are often defined by a "stopword list" which typically consists of about 400 terms including articles, prepositions, conjunctions and certain high-frequency words (some verbs, adverbs and adjectives). Stemming is a technique to reduce words to their grammatical roots so that they can be represented with a only term. A stemmer for English, for example, should identify the string "cats" (and possibly "catlike", "caty", etc.) as based on the root "cat", and "stemmer", "stemming", "stemmed" as based on "stem". Although the presence of sophisticated algorithms, the most popular stemmer is that by Porter [5].

Previously to apply stopword and stemming techniques it is necessary to carry out a lexical analysis process of the documents. Lexical analysis, also named *tokenising*, involves dividing the content of each text into strings of characters called *tokens*.

A suitable tokenising scheme can enhance meaningfully accuracy in text classification. In the most of the filtering systems, a token (also named *feature*) is considered as a string of characters delimited by punctuation symbols, blanks or tabulators.

E-mail classification to spam filtering cannot be considered a traditional text classification problem. Junk e-mails have a "special" structure and content since spammers often introduce "noise" in their messages using phrases like "MONEY!!", "FREE!!!" or placing special characters into the words like "R-o-l-e'X".

The addition of seemingly nonsensical words is aimed at fooling the anti-spam filters that incorporate content-based techniques [6]. These filters examine incoming e-mail messages and calculate the probability of being spam according to intrinsic properties of them (e.g. message subject or body contents, structure, etc.) [7]. Unlike elementary content filters that simply look for specific words in the text, content-based spam filters evolve according to each user needs, analysing all e-mail to determine what words are common to appear in a user legitimate e-mail and which are not. This process is called training, and results in a highly personalized and efficient filtering system.

In order to train and test content-based filters, it is necessary to build a corpus (or use a publicly available corpus) with spam and legitimate e-mails. Anyway, corpora have to be preprocessed to extract their features. Just like text classification, 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 steps of e-mail feature extraction (including the tokenising scheme) and the later selection of the most representative terms are crucial for the performance of the filter.

In this paper we show the results obtained by different well-known content-based techniques when the preprocessing of the training corpus changes. The selected models for the evaluation are three well-known machine learning approaches (Naïve Bayes [8], boosting trees [9] and Support Vector Machines [10]) and three case-based systems for spam filtering that can learn dynamically (a Cunningham *et al.* system which we call *Cum Odds Rate* [11], its improved version named *ECUE* [12] and our SPAMHUNTING system [13]).

The rest of the paper is organized as follows: Section 2 outlines machine learning and case-based e-mail filters mentioned above, Section 3 describes the public corpus used for the empirical model evaluation and studies several issues related with message representation and feature selection. In Section 4 we present the experiments carried out and, finally, in Section 5 we expose the main conclusions reached.

2 Spam Filtering Techniques

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 for spam filtering. 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 the Naïve Bayes algorithm [8].

Besides bayesian models, boosting techniques and Support Vector Machines (SVM) are also well-known machine learning (ML) techniques used in this field.

Boosting algorithms [9] are based on the use of weak learners. 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. Among different boosting algorithms developed for classification tasks we could highlight Adaboost [14].

SVMs [10] 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. Some implementations of SVM can be found in ML environments such as Waikato Environment for Knowledge Analysis¹ (WEKA) or Yet Another Learning Environment² (YALE). Particularly, WEKA includes the *Sequential Minimal Optimization* (SMO) algorithm which has demonstrated a good trade-off between accuracy and speed (see [15] for details).

By other side, the most recent trends in anti-spam filtering are based on the use of Case-Based Reasoning (CBR) systems. Case-based approaches outperform classical machine learning techniques in anti-spam filtering [12] because they work well for disjoint concepts as spam whereas classical ML models try to learn a unified concept description. Another important advantage of case-based approach is the ease with which it can be updated to tackle the *concept drift* problem in the anti-spam domain [16].

Cunningham *et al.* present in [12] a successful case-based system for anti-spam filtering that can learn dynamically. The system uses a similarity retrieval algorithm based on Case Retrieval Nets (CRN) [17]. This classifier uses unanimous voting to determine whether a new e-mail is spam or not. All the returned neighbours need to be classified as spam in order to classify as spam the new message. Delany *et al.* present in [11] an evolution from this previous system called *ECUE (E-mail Classification Using Examples)* that uses a different feature selection method.

Included in the context of lazy learning, our hybrid system named SPAMHUNTING has been introduced in [13] to accurately solve the problem of spam labelling and filtering. The model 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 message. 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 using 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 Corpus Preprocessing

3.1 Analysing the Available Corpus

In order to carry out a benchmarking of the spam filter models, it is essential to have several collections of emails for training and testing purposes. Some researchers on the spam filtering domain had built their own corpus and shared it with the scientific community.

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

² YALE is available from <http://yale.sourceforge.net>

Despite privacy issues, a large number of corpus like SpamAssassin³, Ling-Spam⁴, DivMod⁵, SpamBase⁶ or JunkEmail⁷ can be downloaded from Internet. Table 1 shows a short description of the publicly available corpus focussing in the spam and legitimate ratio and the distribution form.

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

Corpus version	Legitimate Messages		Spam Messages	
	Downloaded file	Message number	Downloaded file	Message number
2002	20021010_easy_ham.tar.bz2	2552	20021010_spam.tar.bz2	502
	20021010_hard_ham.tar.bz2	251		
	20030228_easy_ham.tar.bz2	2500	20030228_spam.tar.bz2	500
2003	20030228_easy_ham_2.tar.bz2	1400		
	20030228_hard_ham.tar.bz2	250	20030228_spam_2.tar.bz2	1397

In our work we use the SpamAssassin corpora, merging emails from 2002 and 2003 versions. It contains 9332 different messages from January 2002 up to and including December 2003. Since this corpus has not been preprocessed by the author, it can successfully be used to analyse the impact of applying different preprocessing techniques.

3.2 Message Representation

The first step in the corpora preprocessing is the e-mail representation. Each message is usually represented as a vector of weighted terms (features) much as in the vector space model in information retrieval domain [3, 4].

Feature identification can be performed by using a variety of generic lexical tools, primarily by tokenising the e-mail into words. At first glance, it seems to be a simple tokenising task guided by spaces and special symbols as word separators. However, at least the following particular cases have to be considered with care: hyphens, backslashes, punctuation marks and the case of the letters (lower and upper case).

Normally, punctuation marks are removed entirely in the process of lexical analysis. However, as mentioned above, punctuation marks and hyphenated words are among the best discriminating attributes in the spam domain because they are more common in spam messages than legitimate ones. Supported by the experiments showed in the following section and the studies carried out by other authors [18], we think that certain punctuations like exclamation, interrogation and hyphens must be part of a term. Also special terms such as *url*, *ip address* should not be splitted but treated as single terms because phrases like "*click http://xxx.xxx.com*" have a high frequency in spam messages. Finally, the case of letters is usually not important for the identification of index terms. As a result, the lexical analyser normally converts all the text to either lower or upper case.

When the tokenising step has been completed, stopwords removal and/or stemming can be applied to identified tokens to reduce the number of representative terms of each e-mail [4].

Once carried out the lexical analysis over the whole corpus, the weight of each term in each message 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 about the total number of occurrences of terms in it, or (iii) TF-IDF where IDF means *Inverse Document Frequency* denoting those terms that are common across the messages of the collection [3].

Usually, the number of terms obtained by the preprocessing stage is very high and the filtering algorithms cannot work with them. Therefore, a last step in the preprocessing is carried out to reduce this number selecting the most representative features. Although some measures exist, *Information Gain* (IG) [19] has been successfully used for aggressive feature removal in several research works [7, 11]. This method is based on computing the IG measure for each identified feature t by using the equation given in Expression (1), where $c \in \{l, s\}$ (legitimate and spam categories, respectively), and selecting those terms having the highest score.

$$IG(t) = \sum_{c \in \{l, s\}} P(t \wedge c) \cdot \log \frac{P(t \wedge c)}{P(t) \cdot P(c)} \quad (1)$$

4 Evaluation

In this work we are trying to find out the impact of applying several basic preprocessing techniques over the corpus before the training and testing stages in the spam filtering domain. Moreover, we are also interested in a comparative study of the feature extraction processes. Therefore, we had compared two different tokenising schemas based on using different token separator characters: (i) [blank:] and [punct:] and (ii) only [blank:]. The first one is based on using blank chars (space, tabs and carriage return marks) and punctuation marks as token separators. In the second approach, we had tried to preserve noise in spam e-mails by using only blank chars as token separators.

Besides of the above tokenising schemas, an implementation of the Porter stemming algorithm [5] and a stopword removal process have been combined and used with Naïve Bayes, SVM, Adaboost and the three previously commented CBR systems (ECUE, *Cum Odds Rate* and SPAMHUNTING). The experiments entail measuring the performance of the analysed models combined with the proposed tokenising schemas over three different scenarios of stemming and stopword removal: (i) applying stopword removal and stemming analysis, (ii) applying stopword but without stemming and (iii) without applying neither stopword nor stemming.

Six well-known metrics [7] have been used in order to evaluate the performance of all the analysed models: *total cost ratio* (TCR) with three different cost values, spam *recall*, spam *precision*, percentage of correct classifications (%OK), percentage of False Positives (%FP) and percentage of False Negatives (%FN). 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.

For our comparisons we have selected the best performance configuration of each technique varying between 100 and 2000 selected features. In order to test the *Cum*

³ Available at <http://www.spamassassin.org/publiccorpus/>

⁴ Available at <http://www.itl.demokritos.gr/>

⁵ Available at <http://www.divmod.org/cvs/corpus/spam/>

⁶ Available at <http://www.ics.uci.edu/~mlearn/MLRepository.html>

⁷ Available at <http://cig.wlv.ac.uk/projects/junk-e-mail/>

Odds Rate and the SPAMHUNTING models we have maintained their original feature selection methods. The first one uses an odds-ratio algorithm described in [12] in order to compute a feature selection having 30 words for representing spam class and 30 words representing legitimate category.

SPAMHUNTING feature selection is carried out in an independent form for each message. The relevant feature list of each message is computed as the minimum set containing the most frequent terms of the specified e-mail, which frequency amount is greater than a specified threshold in the range [0,1]. We are currently using 30% frequency amount as previous works have shown to be the most suitable configuration for achieving the best performance [13].

4.1 Experimental Results

Firstly, the performance of the analysed models is measured from a cost-sensitive point of view in the three predefined scenarios combined with the proposed tokenising schemas. For this purpose, we compute the TCR metric in the above mentioned different situations. TCR assumes than FP errors are λ times more costly than FN errors, where λ depends on the usage scenario (see [7] for more details). In the experiments carried out in this paper the values for λ parameter were 1, 9 and 999.

Table 2 shows the TCR scores comparative of the analysed models, tokenising schemas and preprocessing scenarios. The number of selected features used for each model is placed into square brackets. Results show that preserve noise can help to get better performance in almost all models. However, SVM model works better when noise has been removed. Case-based approaches get the highest correctly classified amount when noise is not removed and stopword removal is used (Scenario 2). Table 2 also shows that noise removal is sometimes useful to get high security (TCR $\lambda=999$).

Table 2. TCR scores over 10 stratified fold-cross validation

Tokenising separator chars	Scenario 1		Scenario 2		Scenario 3	
	[blank:] [punct:]	[blank:]	[blank:] [punct:]	[blank:]	[blank:] [punct:]	[blank:]
Naïve Bayes [1000]	TCR $\lambda=1$ 2.612	2.985	2.609	2.863	2.943	3.174
	TCR $\lambda=9$ 0.501	0.568	0.501	0.508	0.566	0.560
AdaBoost [700]	TCR $\lambda=999$ 0.005	0.006	0.005	0.005	0.006	0.005
	TCR $\lambda=1$ 4.550	4.659	4.403	5.003	5.153	4.854
SVM [2000]	TCR $\lambda=9$ 1.640	1.689	1.578	1.618	1.494	1.432
	TCR $\lambda=999$ 0.021	0.022	0.020	0.019	0.017	0.017
Cum Odds Rate [60]	TCR $\lambda=1$ 27.771	19.011	20.677	17.990	27.423	23.120
	TCR $\lambda=9$ 6.443	3.353	4.257	3.559	6.088	4.920
ECUE [700]	TCR $\lambda=999$ 0.068	0.033	0.043	0.037	0.064	0.050
	TCR $\lambda=1$ 1.308	1.174	1.296	1.364	1.305	1.247
SPAM HUNTING [-]	TCR $\lambda=9$ 1.296	1.153	1.285	1.351	1.294	1.242
	TCR $\lambda=999$ 1.082	0.779	1.074	1.130	1.083	1.140
Naïve Bayes [1000]	TCR $\lambda=1$ 4.441	4.562	4.699	6.020	4.226	5.622
	TCR $\lambda=9$ 2.432	3.071	2.496	2.846	2.265	2.498
AdaBoost [700]	TCR $\lambda=999$ 0.059	0.099	0.052	0.046	0.042	0.036
	TCR $\lambda=1$ 3.903	6.024	3.958	7.498	4.218	4.594
SVM [2000]	TCR $\lambda=9$ 3.493	4.823	3.384	5.331	2.904	2.982
	TCR $\lambda=999$ 2.437	1.545	1.681	0.874	0.499	0.081

From another point of view, Table 3 shows the recall and precision scores obtained for each considered experimental situation. Analysing recall results, and keeping in mind maximizing the highest correctly classified amount, we can realize that classical ML models should not use stopword removal nor stemming (Scenario 3). However, if the same goal is required, CBR/IBR approaches should use stopword removal but not stemming (Scenario 2). Precision scores clearly shows that Naïve Bayes, SVM and Cum Odds Rate get the highest security level (minimize FP errors) when stemming and stopword removal are not used. Nevertheless, these preprocessing techniques should be applied when AdaBoost, ECUE or SPAMHUNTING are used.

Table 4 shows the amount of correct classifications, false positives and false negatives belonging to the experimental work with the six analysed models over the

Table 3. TCR scores over 10 fold-cross validation

Tokenising separator chars	Scenario 1		Scenario 2		Scenario 3	
	[blank:] [punct:]	[blank:]	[blank:] [punct:]	[blank:]	[blank:] [punct:]	[blank:]
Naïve Bayes [1000]	Recall 0.817	0.842	0.837	0.853	0.818	0.869
	Precision 0.801	0.824	0.801	0.807	0.823	0.825
AdaBoost [700]	Recall 0.832	0.836	0.827	0.834	0.865	0.858
	Precision 0.939	0.943	0.938	0.939	0.934	0.928
SVM [2000]	Recall 0.976	0.978	0.976	0.977	0.979	0.916
	Precision 0.979	0.968	0.975	0.964	0.983	0.978
Cum Odds Rate [60]	Recall 0.235	0.150	0.233	0.266	0.228	0.198
	Precision 0.997	0.986	0.996	0.997	0.996	0.998
ECUE [700]	Recall 0.798	0.793	0.787	0.837	0.805	0.846
	Precision 0.969	0.981	0.969	0.971	0.966	0.967
SPAM HUNTING [-]	Recall 0.748	0.837	0.776	0.871	0.753	0.795
	Precision 0.993	0.992	0.991	0.991	0.979	0.978

Table 4. Mean value of correct classifications, FPs and FNs with 10 fold-cross validation

Tokenising separator chars	Scenario 1		Scenario 2		Scenario 3	
	[blank:] [punct:]	[blank:]	[blank:] [punct:]	[blank:]	[blank:] [punct:]	[blank:]
Naïve Bayes [1000]	OK 841.2	852.5	841.4	849.6	857.9	857.9
	False Positives 48.5	43	48.5	48.6	42.8	44
AdaBoost [700]	False Negatives 43.5	37.7	43.3	35	38.9	31.3
	OK 880.3	881.9	878.8	885.1	886.4	883.4
SVM [2000]	False Positives 13	12.2	13.1	14.6	16	16
	False Negatives 39.9	39.1	41.3	34.7	32.2	33.8
Cum Odds Rate [60]	OK 922.7	920.2	921.5	919.1	924.1	922.2
	False Positives 5.9	7.8	6	8.7	4.1	5.3
ECUE [700]	False Negatives 5.6	5.2	5.7	5.4	5	5.7
	OK 750.8	730.4	749.2	758.3	750.4	742.1
SPAM HUNTING [-]	False Positives 0.2	0.5	0.2	0.2	0.2	0.1
	False Negatives 182.2	202.3	183.8	174.7	182.6	191
Naïve Bayes [1000]	OK 878.9	880.2	880.6	893	875.8	889.8
	False Positives 6.2	3.6	6.2	6.1	6.7	6.8
AdaBoost [700]	False Negatives 48.1	49.4	46.4	31.4	50.7	36.6
	OK 871.9	892.9	872.7	900.8	875.8	880.3
SVM [2000]	False Positives 1.2	1.5	1.6	1.8	4	4.2
	False Negatives 60.1	38.8	58.9	30.6	53.4	48.7

defined scenarios and tokenising schemas. Analysing Table 4 we can realize that re-moving noise on tokenising is advisable for achieving the highest amount of correctly classified messages with Naïve Bayes and Case-based models. Adaboost and SVM get the highest amount of correctly classified messages when noise is not removed.

Table 4 also shows that the minimum amount of FP errors using ECUE, *Cum Odds Rate* or Adaboost models is achieved when noise is removed in the tokenising stage. In the other hand, Naïve Bayes, SVM and SPAMHUNTING models work with a highest level of security when noise is not removed.

5 Conclusions and Further Work

In this work we perform a deep analysis of tokenising, stopword removal and stemming techniques. We had chosen a representative set of spam filtering models (including Naïve Bayes, Adaboost, SVM and three case-based systems) in order to see the performance when the context of tokenising, stopword removal and stemming changes.

Table 5. Recommended tokenising, stopword removal and stemming configurations

Tokenising separator chars	Scenario 1		Scenario 2		Scenario 3	
	[:blank] [:punct]	[:blank] [:punct]	[:blank] [:punct]	[:blank] [:punct]	[:blank] [:punct]	[:blank] [:punct]
Naïve Bayes	Maximize hits Minimize FP				✓	✓
Adaboost	Maximize hits Minimize FP	✓			✓	✓
SVM	Maximize hits Minimize FP				✓	✓
<i>Cum Odds Rate</i>	Maximize hits Minimize FP		✓			✓
ECUE	Maximize hits Minimize FP	✓				
SPAM HUNTING	Maximize hits Minimize FP	✓				

Table 5 summarizes the recommended preprocessing context for achieving two different goals: (i) get the maximum amount of correctly classified messages and (ii) get highest security level. As we can see from results, stopword removal is useful for achieving the highest amount of correctly classified messages when case-based reasoning systems are used. This fact represents a weakness of the IG and Odds Ratio feature selection methods as they select some unhelpful stopwords. This happens because the presence of stopwords hamper the selection of features which are really able to differentiate between spam and legitimate messages.

However, although stopwords are semantically void, some of them (such as "however" or "therefore") are unusual in spam messages and should not be filtered. In this way we can highlight the unexpectedly performance of SVM when stemming and/or stopword removal is not used as preprocessing techniques. This happens because un-SVM transforms the input space into a new linearly separable one by removing unhelpful features but keeping useful stopwords. SVM model has its own implicit feature selection method.

The tokenising schema constitutes a relevant issue in the context of spam filtering because, selecting the right choice contributes to improve performance. Keeping in mind the closely relationship between tokenising and classifier, we think that new tokenising schemas (able to remove or preserve noise data) should be proposed, analysed and studied in order to improve the performance of the existing models.

Besides tokenising, stemming and stopword removal issues, we are also interested in analysing the impact of several well-known feature selection methods commonly used in the current domain. We think that modern and classical classifier models are good enough for achieving accuracy results. We could try to improve them or probe new spam filtering models. However, instead this, we are wondering what would happen if we feed current models with an accuracy representation of the training and test messages. Thinking in future work, new and more accuracy techniques of representing the input messages can boost the spam filtering world.

References

1. van Rijsbergen, C.J.: Information Retrieval, 2nd Ed. Butterworths, (1979)
2. Salton, G.: Automatic text processing: The transformation, analysis, and retrieval of information by computer. Addison-Wesley, (1989)
3. Salton, G., McGill, M.: Introduction to Modern Information Retrieval, McGraw-Hill, (1983)
4. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley, (1999)
5. Porter, M.F.: An algorithm for suffix stripping. In Readings in Information Retrieval, (1997) 313-316
6. Gard, D.W.: The state of the art in text filtering. User Modeling and User-Adapted Interaction, Vol.7, (1997) 141-178
7. Androuloupoulos, I., Paliouras, G., Michalakakis, E.: Learning to Filter Unsolicited Commercial E-Mail. Technical Report 2004/2, NCSR "Demokritos", (2004)
8. Saharni, 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, Technical Report WS-98-05, (1998) 55-62
9. Carreras, X., Marquez, 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
10. Vapnik, V.: The Nature of Statistical Learning Theory, 2nd Ed. Statistics for Engineering and Information Science, (1999)
11. 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
12. 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)
13. Fdez-Riverola, F., Méndez, J.R., Iglesias, E.L., Diaz, F. Representación Flexible de e-mails para la construcción de filtros antispam: un caso práctico. Proc. of the I Congreso Español de Informática CEDT05 (2005) 109-116

14. Schapire, R.E., Singer, Y.: Boostexter: a boosting-based system for text categorization. *Machine Learning*, Vol. 39 (2/3), (2000) 135–168
15. 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
16. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. *Machine Learning*, Vol. 23 (1), (1996) 69–101
17. Lenz, M., Aurio, E., Manago, M.: Diagnosis and Decision Support. *Case-Based Reasoning Technology. Lecture Notes in Artificial Intelligence*, Vol. 1400, (1998) 51–90
18. Le, Z., Tian-shun, Y.: Filtering Junk Mail with A Maximum Entropy Model. *Proc. of the ICCPOL2003*, (2003) 446–453
19. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. *Proc. of the Fourteenth International Conference on Machine Learning. ICMML-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

Toward a Motivated BDI Agent Using Attributes Embedded in Mental States

José Cascalho¹, Luis Antunes¹, Milton Corrêa², and Helder Coelho¹

¹ Departamento de Informática

Faculdade de Ciências da Universidade de Lisboa

Bloco C6, Piso 3, Campo Grande

1749-016 Lisboa, Portugal

{jmc@notes.uac.pt, xarax@di.fc.ul.pt, hcoelho@di.fc.ul.pt}

² Coordenação de Informática

Laboratório Nacional de Cálculo Científico

Av. Gertúlio Vargas, 333

Petrópolis, RJ-Brasil

mcorrea@lnc.br

Abstract. In this paper we discuss how to apply the Mental State Framework to model an agent that uses different motivations to make decisions. We employ several mental states attributes to characterize the mechanism of decision and we add a set of motivation-derived desires to the top of a BDI-like architecture. We show that an agent can change her behavior if motives behind the decisions change. Finally we explore reasons behind reasons, arguing that we can expect to identify different agent types. Endowed with such operational extensions, the Mental State Framework can provide a tool with which to program flexible agents in a principled way.

1 Introduction

Let us suppose we need to buy plane tickets and we do it with the help of software agents. We decide to buy a ticket to a location where we have a meeting in a specific date, to a maximum price of 350 Euro and we want to go on a direct flight. If we search for a ticket using an interface belonging to a specific software house, the agents, indirectly by the users, are ranked by its performance. It is in the interest of the software house to get the best performance as possible.

Suppose agents have a measure of importance that evaluates the cost/benefit ratio toward decisions related with agents' goals, i.e., the goal to return with an answer (positive or negative) and the goal to do the search, and also have a measure of urgency which evaluates the time left for agents to fulfill that goal. As the deadline approaches, the urgency for returning will surpass the importance of searching even if the goal is not yet reached.

If an agent returns too soon, he may risk to loose for another agent returning later with a better offer. On the other hand, if an agent returns too late, the user might already have decided to choose another agent's proposal. What should be the best strategy for a software house to be successful in search?