Twitter Content-based Spam Filtering

Igor Santos, Igor Miñambres-Marcos, Carlos Laorden, Patxi Galán-García, Aitor Santamaría-Ibirika, and Pablo G. Bringas

DeustoTech-Computing, Deusto Institute of Technology (DeustoTech)
Avenida de las Universidades 24, 48007 Bilbao, Spain
isantos@deusto.es, igor.m@opendeusto.es, claorden@deusto.es, patxigg@deusto.es, a.santamaria@deusto.es, pablo.garcia.bringas@deusto.es

Abstract. Twitter has become one of the most used social networks. And, as happens with every popular media, it is prone to misuse. In this context, spam in Twitter has emerged in the last years, becoming an important problem for the users. In the last years, several approaches have appeared that are able to determine whether an user is a spammer or not. However, these blacklisting systems cannot filter every spam message and a spammer may create another account and restart sending spam. In this paper, we propose a content-based approach to filter spam tweets. We have used the text in the tweet and machine learning and compression algorithms to filter those undesired tweets.

Keywords: spam filtering; Twitter; social networks; machine learning; text classification

1 Introduction

Similarly as happened with e-mail, online social networks have emerged and become a powerful communication media, where users can share links, discuss and connect with each other. In particular, Twitter is one of the most used social networks, a very popular social system that serves also as a powerful communication channel, a popular news source; and as happens with every powerful channel, it is prone to misuse and, therefore, a spam marketplace has been created.

This new type of spam has several particularities (such as the limitation of 140 characters) and, despite the Twitter's effort to combat these spam operations, there is still no a proper solution to this problem [1]. And it is known that spam is not only very annoying to every-day users, but also constitutes a major computer security problem that costs billions of dollars in productivity losses [2]. It can also be used as a medium for phishing (i.e., attacks that seek to acquire sensitive information from end-users) [3] and the spread of malicious software (e.g., computer viruses, Trojan horses, spyware and Internet worms) [2].

Regarding Twitter spam filtering, there has been an increment in the research activity in the last years. In particular, several methods have appeared to detect spammer accounts in Twitter. Benevenuto et al. [4] developed a system to detect spammers in Twitter by using the properties of the text of the users'

tweets and user behavioural attributes, being able to detect about a 70% of the spammer accounts. Grier et al. [5] proposed a schema based on URL blacklisting. Wang [6] proposed a new approach that employed features from the social graph from Twitter users to detect and filter spam. Gao et al. [7] presented a generic Online Social Network spam filtering system that they tested using Facebook and Twitter data. Their approach used several features using clustering to detect spam campaigns. Similarly, Ahmed and Abulaish [8] presented a new approach to identify spammer accounts using classic statistical methods.

Although these approaches are able to deal with the problem, a new spammer account may emerge to substitute the filtered accounts. Hence, these blacklisting systems, similarly as happens with e-mails, should be complemented with content-based approaches commonly used in e-mail spam filtering. In this context, a recent work by Martinez-Romo and Araujo proposed a method for detecting spam tweets in real time using different language models and measuring the divergences [9].

Against this background, we present here a study of how classic e-mail content-based techniques can filter spam in Twitter. To this end, we have comprised a dataset (publicly available) of spam and ham (not spam) tweets. Using these data, we have tested several content-based spam filtering methods. In particular, we have tested statistical methods based on the bag of word models, and also compression-based text classification algorithms.

In summary, we advance the state of the art through the following contributions: (i) a new and public dataset of Twitter spam, to serve as evaluation of Twitter spam filtering systems; (ii) we adapt content-based spam filtering to Twitter; (iii) we present a new compression-based text filtering library for the well-known machine-learning tool WEKA; and (iv) we show that the proposed method achieves high filtering rates, even on completely new, previously unseen spam, discussing the weakness of the proposed model and explain possible enhancements.

The remainder of this paper is organised as follows. Section 2 explains the methods used in this study. Section 3 details the process to build the dataset, the experiments performed and presents the results. Section 4 discusses the main shortcomings of the proposed method and proposes possible improvements, outlining avenues for future work.

2 Content-based Spam Filtering Methods

In order to find a solution to the problem of spam, the research community has undertaken a huge amount of work. Because *machine learning* approaches have succeeded in text categorisation problems [10], these techniques have been adopted in spam filtering systems. Consequently, substantial work has been dedicated to the *Naïve Bayes* classifier [11], with studies in anti-spam filtering confirming its effectiveness [12–14]. Another broadly embraced machine-learning-based technique is *Support Vector Machines* (SVM) [15]. The advantage of SVM is that its accuracy does not degrade even with many features [16]. Therefore,

such approaches have been applied to spam filtering [17, 18]. Likewise, *Decision Trees* that classify by means of automatically learned rule-sets [19], have also been used for spam filtering [20]. All of these machine-learning-based spam filtering approaches are termed *statistical approaches* [21].

Machine learning approaches model e-mail messages using the *Vector Space Model* (VSM) [22], an algebraic approach for *Information Filtering* (IF), *Information Retrieval* (IR), indexing and ranking. This model represents natural language documents in a mathematical manner through vectors in a multidimensional space.

However, this method has its shortcomings. For instance, in spam filtering, *Good Words Attack* is a method that modifies the term statistics by appending a set of words that are characteristic of legitimate e-mails. In a similar vein, *tokenisation attacks* work against the feature selection of the message by splitting or modifying key message features rendering the term-representation no longer feasible [23]. Compression-based text classification methods have been applied for spam filtering [2], with good results solving this issue.

In the remainder of this section, we detail the methods we have employed.

2.1 Machine-learning Classification

Machine-learning is an active research area within *Artificial Intelligence* (AI) that focuses on the design and development of new algorithms that allow computers to reason and decide based on data (i.e. computer learning). We use supervised machine-learning; however, in the future, we would also like to test unsupervised methods for spam filtering. In the remainder of this section, we review several supervised machine-learning approaches that have succeeded in similar domains.

- **Bayesian Networks:** Bayesian Networks [24] are based on *Bayes' Theorem* [25]. They are defined as graphical probabilistic models for multivariate analysis. Specifically, they are directed acyclic graphs that have an associated probability distribution function [26]. Nodes within the directed graph represent problem variables (they can be either a premise or a conclusion) and the edges represent conditional dependencies between such variables. Moreover, the probability function illustrates the strength of these relationships in the graph [26].
- **Decision Trees:** These models are a type of machine-learning classifiers that are graphically represented as trees. Internal nodes represent conditions regarding the variables of a problem, whereas final nodes or leaves represent the ultimate decision of the algorithm [19]. Different training methods are typically used for learning the graph structure of these models from a labelled dataset. We used *Random Forest*, an ensemble (i.e., combination of weak classifiers) of different randomly-built decision trees [27], and *J48*, the WEKA [28] implementation of the *C4.5* algorithm [29].
- **K-Nearest Neighbour:** The *K-Nearest Neighbour* (KNN) [30] classifier is one of the simplest supervised machine learning models. This method

classifies an unknown specimen based on the class of the instances closest to it in the training space by measuring the distance between the training instances and the unknown instance. Even though several methods to choose the class of the unknown sample exist, the most common technique is to simply classify the unknown instance as the most common class amongst the K-nearest neighbours.

– **Support Vector Machines (SVM):** SVM algorithm divide the data space into two regions using a *hyperplane*. This hyperplane always maximises the *margin* between those two regions or classes. The margin is defined by the farthest distance between the examples of the two classes and computed based on the distance between the closest instances of both classes, which are called *supporting vectors* [15]. Instead of using linear hyperplanes, it is common to use the so-called *kernel functions*. These kernel functions lead to non-linear classification surfaces, such as polynomial, radial or sigmoid surfaces [31].

2.2 Compression-based Text Classifier

In this section, we describe how we have performed the text classification using the compression models. In particular, given a set of training documents $\mathcal{D}$, we can generate a compression model $\mathcal{M}$ using these documents.

The training documents are previously labelled in n different classes, dividing our training documents $\mathcal{D}$ in n different document sets $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_{n-1}, \mathcal{D}_n)$ depending on their labelled class (in our task 2: spam or ham, that is $\mathcal{D}_{spam}$ and $\mathcal{D}_{ham}$)). In this way, we generate n different compression models $\mathcal{M}$, one for each class: $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_{n-1}, \mathcal{M}_n)$.

When a training document $d_{i,j}$, where i denotes the class and j denotes the document number, is analysed, we add it to the compression model of its class $\mathcal{M}_i$, training and updating the compression model. In this way, if we proceed with the training of every document in $\mathcal{D}$, each model will be trained with documents of that class, therefore being adapted and prepared to compress only documents of that particular class.

In particular, for the compression models used, each model is given a training sequence in order to learn the model $\mathcal{M}$ that provides each future outcome a probability value for each future symbol. The prediction performance is usually measured by the average log-loss [32] $\ell(\hat{\mathcal{M}}, x_1^T)$ that given a test sequence $d_1^T = (d_1, d_2, ..., d_{n-1}, d_n)$:

$$\ell(\hat{M}, x_1^T) = \frac{1}{T} \log \hat{\mathcal{M}}(d_i | d_1, d_2, ..., d_{i-1}) \tag{1}$$

A small average log-loss over a test sequence means that the sequence is going to be well compressed. In this way, the mean of the log-loss of each model $\mathcal{M}$ achieves the best possible entropy:

$$LogEval(d, \mathcal{M}) = \frac{1}{T} \sum_{i=1}^{T} \log \mathcal{M}(d_i | d_1^{i-1}) \tag{2}$$

In this case, it is also possible to adapt the model with the symbols d_i of the test sequence, generating another way of measuring considering the sample of the type of data generated by the model:

$$LogEval(d, \mathcal{M}) = \frac{1}{T} \sum_{i=1}^{T} \log \mathcal{M}'(d_i | d_1^{i-1})) \tag{3}$$

where M' is the model adapted with the d_{i-1} symbol.

In both cases, the class of tested document is selected as the minimal value of each compression model tested. When a testing document d arrives, our system can also evaluate the probability of a document belonging to a class using each compression model $\mathcal{M}_i$ and then we generate a probability with values between 0 and 1, using the next formula:

$$P(d \in \mathcal{M}_i) = \frac{\frac{1}{LogEval(d, \mathcal{M}_i)}}{\sum_{j=1}^{n} \frac{1}{LogEval(d, \mathcal{M}_i)}} \tag{4}$$

Similarly, we may use the minimum value of $LogEval$:

$$\mathcal{M}_i(d) = \underset{\mathcal{M}_i \in \mathcal{M}}{\operatorname{argmin}} \, LogEval(d, \mathcal{M}_i) \tag{5}$$

Using the implementation of Prediction by Partial Matching (PPM) [33], Lempel-Ziv 78 (LZ78) [34], an improved LZ78 algorithm (LZ-MS) [35], Binary Context Tree Weighting Method (BI-CTW) [36], Decomposed Context Tree Weighting (DE-CTW) [37], Probabilistic Suffix Trees (PST) [38], presented in [32][1] and adapting the original version of DMC [39][2], we have adapted these models and provided a WEKA[28] 3.7 package[3] named `CompressionTextClassifier` used in this paper.

We briefly describe the 2 compression models used for Twitter spam filtering. We used DMC and PPM because they have been the most used ones in e-mail spam filtering:

– **Dynamic Markov Chain (DMC):** DMC [39] models information with a finite state machine. Associations are built between every possible symbol in the source alphabet and the probability distribution over those symbols. This probability distribution is used to predict the next binary digit. The DMC method starts in an already defined state, changing the state when new bits are read from the entry. The frequency of the transitions to either 0 or 1 are summed when a new symbol arrives. The structure can also be updated using a state cloning method. DMC has been previously used in e-mail spam filtering tasks [2] and in SMS spam filtering [40, 41] obtaining high filtering rates.

[1] Available at: `http://www.cs.technion.ac.il/~ronbeg/vmm/code_index.html`

[2] Available at: `http://www.jjj.de/crs4/dmc.c`

[3] Available at: `http://paginaspersonales.deusto.es/isantos/Resources/CompressionTextClassifier-0.4.3.zip`

– **Prediction by Partial Match (PPM):** Prediction by partial matching (PPM) algorithm [33] is one of the best lossless compression algorithms. The implementation is based on a prefix tree [32]. In this way, using the training character string, the algorithm constructs the tree. In a similar vein as the LZ78 prefix tree, each node within the tree represents a symbol and has a counter of occurrences. The tree starts with a root symbol node for an empty sequence. It parses each symbol of the training sequence and the parsed symbol and its context, allowing the model to define a potential path in the tree. Once the tree is generated, the resultant data structure can be used to predict each symbol and context, transversing the tree according to the longest suffix of the testing sequence.

3 Evaluation

To evaluate the proposed method, we constructed a dataset comprising 31,457 tweets retrieved from a total of 223 user accounts, from which 143 were spam accounts verified in the TweetSpike site [4] while 80 were randomly selected among legitimate users. Besides, the accounts were manually checked to determine if they were correctly classified. Regarding the tweets, after removing all the duplicated instances, the final dataset used for evaluation of the proposed algorithms had 25,846, from which 11,617 correspond to spam tweets and 14,229 to legitimate tweets[5].

– **Cross validation:** In order to evaluate the performance of machine-learning classifiers, *k-fold cross validation* is commonly used in machine-learning experiments.
  For each classifier tested, we performed a k-fold cross validation with $k = 10$. In this way, our dataset was split 10 times into 10 different sets of learning sets (90% of the total dataset) and testing sets (10% of the total data).
– **Learning the model:** For each fold, we performed the learning phase for each of the algorithms presented in Section 2.1 with each training dataset, applying different parameters or learning algorithms depending on the concrete classifier. If not specified, the default ones in WEKA were used.
  To evaluate each classifier's capability we measured *accuracy*, which is the total number of the classifier's hits divided by the number of messages in the whole dataset (shown in equation 6).

$$Accuracy(\%) = \frac{TP + TN}{TP + FP + FN + TN} \cdot 100 \tag{6}$$

  where TP is the amount of correctly classified spam (i.e., true positives), FN is the amount of spam misclassified as legitimate mails (i.e., false negatives),

[4] Available at: `http://www.tweetspike.org`
[5] Available online at: `http://paginaspersonales.deusto.es/isantos/resources/twitterspamdataset.csv`

FP is the amount of legitimate mail incorrectly detected as spam, and TN is the number of legitimate mail correctly classified.
Furthermore, we measured the precision of the spam identification as the number of correctly classified spam e-mails divided by the number of correctly classified spam e-mails and the number of legitimate e-mails misclassified as spam:

$$S_P = \frac{N_{s\to s}}{N_{s\to s} + N_{l\to s}} \tag{7}$$

where $N_{s\to s}$ is the number of correctly classified spam messages and $N_{l\to s}$ is the number of legitimate e-mails misclassified as spam.
Additionally, we measured the recall of the spam e-mail messages, which is the number of correctly classified spam e-mails divided by the number of correctly classified spam e-mails and the number of spam e-mails misclassified as legitimate:

$$S_R = \frac{N_{s\to s}}{N_{s\to s} + N_{s\to l}} \tag{8}$$

We also computed the F-measure, which is the harmonic mean of both the precision and recall, as follows:

$$F\text{-}measure = \frac{2N_{s\to s}}{2N_{s\to s} + N_{s\to l} + N_{l\to s}} \tag{9}$$

Finally, we measured the *Area Under the ROC Curve* (AUC), which establishes the relation between false negatives and false positives. The ROC curve is represented by plotting the rate of true positives (TPR) against the rate of false positives (FPR).

Table 1 shows the results obtained with both the common machine-learning algorithms and the compression models proposed. On the one hand, we can appreciate how one of the most used classifiers in e-mail spam filtering, Naive Bayes, performs poorly when compared to the rest of the models, obtaining a 0.76 of AUC. In a similar vein, PPM with adaptation obtains a 0.86 of AUC against the 0.92-0.99 of the rest of the classifiers. Later SVM, with different kernels, and KNN algorithms, obtain a lowest value of AUC of 0.92 for SVM with Radial Basis Function and a highest value of AUC of 0.97 for KNN with a k of 3. The decision tree C4.5 also obtains a 0.97 of AUC but improving every other measure when compared to the previous algorithms. On the next scale, there are algorithms with 0.98-0.99 of AUC. Amongst them, Random Forest are the ones with the best behaviour, obtaining not only a 0.99 of AUC but also significant results in the other measures. The only algorithms with the same performance are DMC and PPM, both without adaptation, that also obtain a 0.99 of AUC. Finally, with 0.98 of AUC we can find the rest of the Bayes-based algorithms and DMC with adaptation.

Table 1. Results of the evaluation of common machine learning classifiers and the compression models.

Classifier	*Acc.*	S_P	S_R	$F-Measure$	*AUC*
Random Forest N=50	96.42	0.98	0.94	0.96	0.99
Random Forest N=30	96.41	0.98	0.94	0.96	0.99
DMC without Adaptation	95.99	0.96	0.95	0.96	0.99
Random Forest N=10	95.96	0.97	0.94	0.95	0.99
PPM without Adaptation	94.80	0.97	0.91	0.94	0.99
Naive Bayes Multinomial Word Frequency	94.94	0.95	0.93	0.94	0.98
Naive Bayes Multinomial Boolean	94.75	0.95	0.93	0.94	0.98
Bayes K2	94.12	0.99	0.88	0.93	0.98
DMC with Adaptation	93.11	0.94	0.90	0.92	0.98
C4.5	95.79	0.98	0.92	0.95	0.97
KNN K=3	93.71	0.97	0.89	0.93	0.97
KNN K=5	92.61	0.97	0.86	0.91	0.97
SVM PVK	95.81	0.97	0.93	0.95	0.96
KNN K=1	94.22	0.95	0.92	0.93	0.96
SVM Lineal	94.38	0.92	0.96	0.94	0.95
SVM RBF	93.20	0.99	0.85	0.92	0.92
PPM with Adaptation	76.50	0.78	0.69	0.72	0.86
Naive Bayes	72.72	0.64	0.89	0.75	0.76

4 Conclusions

In this paper, we presented a study of how classic e-mail content-based techniques can filter Twitter spam, adapting the algorithms to Twitter's peculiarities, but, while analysing the text of the messages has proven as a great approach to identifying the type of the communications. In particular: (i) we have compiled a new and public dataset of spam in Twitter, (ii) we evaluated the classic content-based spam filtering techniques to this type of spam, and (iii) as a technical contribution, we have presented a new and free software compression-based text filtering library for the well-known machine-learning tool WEKA.

Future versions of this spam filtering system will move in two main directions. First, we will enhance this approach using social network features. Second, we plan to enhance the semantic capabilities, as already studied in e-mail spam filtering [42, 43], by studying the linguistic relationships in tweets.

References

1. Thomas, K., Grier, C., Song, D., Paxson, V.: Suspended accounts in retrospect: an analysis of twitter spam. In: Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, ACM (2011) 243–258
2. Bratko, A., Filipič, B., Cormack, G., Lynam, T., Zupan, B.: Spam filtering using statistical data compression models. The Journal of Machine Learning Research **7** (2006) 2673–2698

3. Jagatic, T., Johnson, N., Jakobsson, M., Menczer, F.: Social phishing. Communications of the ACM **50**(10) (2007) 94–100
4. Benevenuto, F., Magno, G., Rodrigues, T., Almeida, V.: Detecting spammers on twitter. In: Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS). (2010)
5. Grier, C., Thomas, K., Paxson, V., Zhang, M.: @spam: The underground on 140 characters or less. In: Proceedings of the 17th ACM conference on Computer and communications security, ACM (2010) 27–37
6. Wang, A.H.: Don't follow me: Spam detection in twitter. In: Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on, IEEE (2010) 1–10
7. Gao, H., Chen, Y., Lee, K., Palsetia, D., Choudhary, A.: Towards online spam filtering in social networks. In: Symposium on Network and Distributed System Security (NDSS). (2012)
8. Ahmed, F., Abulaish, M.: A generic statistical approach for spam detection in online social networks. Computer Communications (2013) in press.
9. Martinez-Romo, J., Araujo, L.: Detecting malicious tweets in trending topics using a statistical analysis of language. Expert Systems with Applications (2012)
10. Sebastiani, F.: Machine learning in automated text categorization. ACM computing surveys (CSUR) **34**(1) (2002) 1–47
11. Lewis, D.: Naive (Bayes) at forty: The independence assumption in information retrieval. Lecture Notes in Computer Science **1398** (1998) 4–18
12. Schneider, K.: A comparison of event models for Naive Bayes anti-spam e-mail filtering. In: Proceedings of the 10^{th} Conference of the European Chapter of the Association for Computational Linguistics. (2003) 307–314
13. Androutsopoulos, I., Koutsias, J., Chandrinos, K., Spyropoulos, C.: An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In: Proceedings of the 23^{rd} annual international ACM SIGIR conference on Research and development in information retrieval. (2000) 160–167
14. Seewald, A.: An evaluation of naive Bayes variants in content-based learning for spam filtering. Intelligent Data Analysis **11**(5) (2007) 497–524
15. Vapnik, V.: The nature of statistical learning theory. Springer (2000)
16. Drucker, H., Wu, D., Vapnik, V.: Support vector machines for spam categorization. IEEE Transactions on Neural networks **10**(5) (1999) 1048–1054
17. Blanzieri, E., Bryl, A.: Instance-based spam filtering using SVM nearest neighbor classifier. Proceedings of FLAIRS-20 (2007) 441–442
18. Sculley, D., Wachman, G.: Relaxed online SVMs for spam filtering. In: Proceedings of the 30^{th} annual international ACM SIGIR conference on Research and development in information retrieval. (2007) 415–422
19. Quinlan, J.: Induction of decision trees. Machine learning **1**(1) (1986) 81–106
20. Carreras, X., Márquez, L.: Boosting trees for anti-spam email filtering. In: Proceedings of RANLP-01, 4th international conference on recent advances in natural language processing, Citeseer (2001) 58–64
21. Zhang, L., Zhu, J., Yao, T.: An evaluation of statistical spam filtering techniques. ACM Transactions on Asian Language Information Processing (TALIP) **3**(4) (2004) 243–269
22. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM **18**(11) (1975) 613–620
23. Wittel, G., Wu, S.: On attacking statistical spam filters. In: Proceedings of the 1^{st} Conference on Email and Anti-Spam (CEAS). (2004)

24. Pearl, J.: Reverend bayes on inference engines: a distributed hierarchical approach. In: Proceedings of the National Conference on Artificial Intelligence. (1982) 133–136
25. Bayes, T.: An essay towards solving a problem in the doctrine of chances. Philosophical Transactions of the Royal Society **53** (1763) 370–418
26. Castillo, E., Gutiérrez, J.M., Hadi, A.S.: Expert Systems and Probabilistic Network Models. Erste edn., New York, NY, USA (1996)
27. Breiman, L.: Random forests. Machine learning **45**(1) (2001) 5–32
28. Garner, S.: Weka: The Waikato environment for knowledge analysis. In: Proceedings of the 1995 New Zealand Computer Science Research Students Conference. (1995) 57–64
29. Quinlan, J.: C4. 5 programs for machine learning. Morgan Kaufmann Publishers (1993)
30. Fix, E., Hodges, J.L.: Discriminatory analysis: Nonparametric discrimination: Small sample performance. technical report project 21-49-004, report number 11. Technical report, USAF School of Aviation Medicine, Randolf Field, Texas (1952)
31. Amari, S., Wu, S.: Improving support vector machine classifiers by modifying kernel functions. Neural Networks **12**(6) (1999) 783–789
32. Begleiter, R., El-Yaniv, R., Yona, G.: On prediction using variable order markov models. J. Artif. Intell. Res. (JAIR) **22** (2004) 385–421
33. Cleary, J., Witten, I.: Data compression using adaptive coding and partial string matching. Communications, IEEE Transactions on **32**(4) (1984) 396–402
34. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory **24**(5) (1978) 530–536
35. Nisenson, M., Yariv, I., El-Yaniv, R., Meir, R.: Towards behaviometric security systems: Learning to identify a typist. In: Knowledge Discovery in Databases: PKDD 2003, Springer (2003) 363–374
36. Willems, F.: The context-tree weighting method: Extensions. Information Theory, IEEE Transactions on **44**(2) (1998) 792–798
37. Volf, P.A.J.: Weighting techniques in data compression: Theory and algorithms. Citeseer (2002)
38. Ron, D., Singer, Y., Tishby, N.: The power of amnesia: Learning probabilistic automata with variable memory length. Machine learning **25**(2) (1996) 117–149
39. Cormack, G., Horspool, R.: Data compression using dynamic markov modelling. The Computer Journal **30**(6) (1987) 541–550
40. Cormack, G., Gómez Hidalgo, J., Sánz, E.: Spam filtering for short messages. In: Proceedings of the 16th ACM conference on Conference on information and knowledge management, ACM (2007) 313–320
41. Cormack, G., Hidalgo, J., Sánz, E.: Feature engineering for mobile(sms) spam filtering. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. Volume 23. (2007) 871–872
42. Santos, I., Laorden, C., Sanz, B., Bringas, P.G.: Enhanced topic-based vector space model for semantics-aware spam filtering. Expert Systems With Applications **39**(1) 437–444 doi:10.1016/j.eswa.2011.07.034.
43. Laorden, C., Santos, I., Sanz, B., Alvarez, G., Bringas, P.G.: Word sense disambiguation for spam filtering. Electron. Commer. Rec. Appl. **11**(3) (May 2012) 290–298