Spam Detection Filter using KNN Algorithm and Resampling

Loredana Firte
Tehnical University of Cluj-Napoca
loredana.firte@gmail.com

Camelia Lemnaru
Tehnical University of Cluj-Napoca
camelia.lemnaru@cs.utcluj.ro

Rodica Potolea
Tehnical University of Cluj-Napoca
rodica.potolea@cs.utcluj.ro

Abstract— **Spamming has become a time consuming and expensive problem for which several new directions have been investigated lately. This paper presents a new approach for a spam detection filter. The solution developed is an offline application that uses the k-Nearest Neighbor (kNN) algorithm and a pre-classified email data set for the learning process.**

Keywords- spam, filter, kNN, resample

I. INTRODUCTION

Spamming - as the abuse of electronic messaging systems - has become a real problem in recent years. Spam is in several ways not harmless. The least it does is taking away bandwidth from the internet users and time to process it. According to studies undertaken by M86 Security, the ascending trend that European IT has lately followed (as compared to North America) involves not only positive aspects; with it came less pleasant phenomena like spam. If until recently America was the leading provider of spam, the European servers have now registered the largest source of spam. The increase in the amount of spam sent by the European servers is actually a natural consequence of the connection's amplified speed.

Email spam refers to sending irrelevant, inappropriate and unsolicited email messages to numerous people [1]. This is possible due to the low entrance barrier and the low cost of sending emails, which makes it one of the most popular forms of spam. The purpose of email spam is advertising, promotion, and spreading backdoors or malicious programs. Currently, Phishing is also considered as one of the main goals of spammers when employing email spams.

This paper presents a new spam detection filter, which combines several attributes, some previously used [2], mainly based on word frequency, and applies the kNN algorithm to classify the emails.

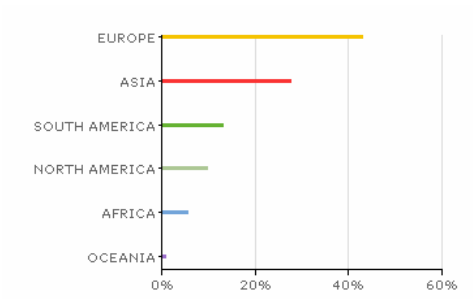


Figure 1. Spam Sources by Continent

The rest of the paper is organized as follows: section II reviews the main related work on spam filtering and existing applications. Section III describes the system's architecture in our approach and its workflow, followed in the next sections by the two components' description: feature extraction and the classification module. Section VI presents the experimental work performed and the results obtained. The last part summarizes this article's conclusions and presents some further work that might be pursued.

II. RELATED WORK

In recent years, anti-spam filters have become necessary tools for Internet service providers to tackle the continuously growing spam phenomenon. Current server-side anti-spam filters are made up of several modules aimed at detecting different features of spam e-mails. We briefly present an overview of the most significant work done in this field.

In 1998 the first published connection between e-mail classification and Naïve Bayes covered the spam filtering project of Stanford researcher Mehran Shami and his Microsoft team. Shami was able to achieve precisions upwards of 97 percent using only word based features [3]. In the following years, research has shown Naïve Bayes to perform better than keyword based spam filters [4]. Nearly all major enterprise level spam filtering software includes naïve Bayesian classification as at least a factor in filtering, and in many very effective solutions it plays the central role. The popular open source spam filtering software D-Spam claims success rates in excess of 99.5 percent using a Bayesian classification algorithm at its core.

The simplest filtering techniques are black-listing and white-listing [5]. A black-list is a list of email addresses and domains that are considered to be used by spammers. All incoming messages from an address or domain that appears in the black-list are automatically considered spam without further processing. Similarly, a white-list represents an index of trusted correspondents and domains, thus any incoming message from the list is automatically accepted. Most email client programs allow manual updates while some also connect to one or more Real-time Black-Lists (RBL).

Another popular way to filter spam is through handcrafted keyword-based rules [6]. These rules attempt to identify spam by looking for specific words or phrases in the message's subject or other suspicious patterns in the header. Modern email clients make such rules easy to write and share, even for novice users.

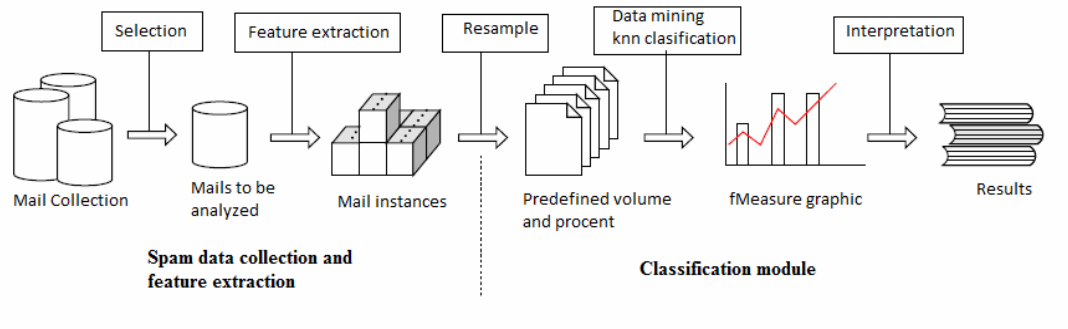


Figure 2. System architecture

In particular, text categorization techniques have been investigated by researchers in order to design modules for the analysis of the semantic content of e-mails. They have potentially higher generalization capability with respect to manually derived classification rules used in current server-side filters. Recently spammers introduced a new trick consisting of embedding the spam message into attached images, which can make ineffective all current techniques based on the analysis of digital text in the subject and body fields of e-mails. In [7] is proposed an approach to antispam filtering which exploits the text information embedded into images sent as attachments based on the application of state-of-the-art text categorization techniques to the analysis of text extracted by OCR tools from images attached to e-mails.

III. SYSTEM ARCHITECTURE

We proposed an architecture divided into two main components: one that collects data and extracts the characteristics of each message and another one that classifies the emails and interprets the results.

As shown in Fig. 2, each component has several processes that have to be ran in this particular order. First, an email selection is done in order to collect data to be analyzed. It is easy to create ad-hoc email addresses and to advertise them in a controlled manner to attract spam. Such addresses are sometimes called “spam traps" and are generally used by email filtering companies to obtain a continuous clean feed of spam for analysis.

The next step is a feature extraction process: the message is analyzed and several attributes (as message and subject length, number of replies, images, links and attachments and word frequencies) are saved. An email and its values for the extracted attributes represent an instance of the data set. From the stream of preprocessed tokens, there will have to be extracted features that provide the classification method with sufficiently discriminative information to allow for a highly accurate decision. Several methods have been developed to extract features from text, especially when the text may include meta contents in header information like email. Different methods lead to different performances. Simple tokenization is most popular, but cannot obtain good results for spammer’s intentional obfuscation or good words attack. In some other languages e.g. Chinese, tokenization itself is a challenging problem. Some filters [8] treat each message as a sequence of characters or bits, and automatically discover patterns of interest. Others match header information and also the implicit semantics of words and other tokens in identifying features. Such filters are best suited to corpora that contain raw messages with complete headers.

Here we combined the tokenization method for the body of the message with a stemming process. This way we do not look for full words repeating, but for words with the same stem We also avoided some of the possible obfuscations by removing text between tags and all characters other then letters.

In [9] it is pointed out that the proportion of spam messages reported in research data sets varies considerably, from 16.6% to 88.2%. This may be simply because the proportion varies significantly from one individual to another. The amount of spam received depends on the email address, the degree of exposure, the amount of time the address has been public and the upstream filtering. The amount of legitimate email received similarly varies greatly from one individual to another.

Furthermore, the sampling strategy performs nearly as well as the strategy that evaluates many different class distributions and chooses the best-performing one (which is optimal in terms of classification performance but inefficient in terms of the number of examples required). For many data sets the class distribution that yields the best performing classifiers remains relatively constant for different training-set sizes, supporting the notion that there often is a “best” marginal class distribution. Results have shown that as the amount of training data increases, the differences in performance for different class distributions diminish. This indicates that as more data becomes available, the choice of marginal class distribution becomes less and less important, especially in the neighborhood of the optimal distribution.

We have performed with our system several evaluations to identify the most appropriate combination between the data set size and class distribution. Results suggested that a resampling strategy for getting a different distribution on the training set might help. Therefore, after emails are processed and transformed in classifying instances, the data set is resampled to an appropriate size

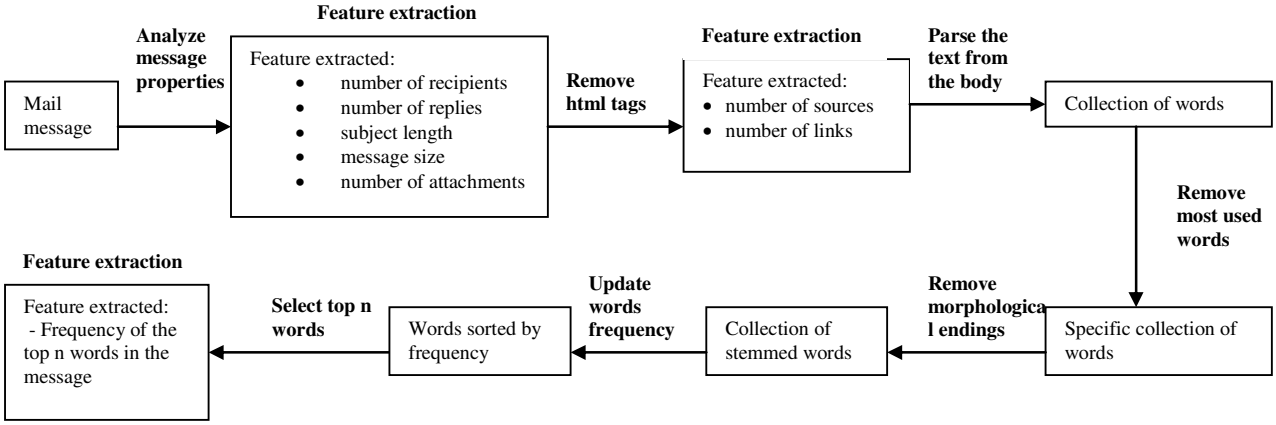


Figure 3. Feature extraction

and bulk distribution. After resampling the data is considered for further classification, and each new occurring instance (email) is classified based on the similarity with existing known instances using a kNN method.

Once an email is analyzed, all its features are stored in order to allow further attributes' update. In the real process of spamming, the email corpus is dynamic so the size and the distribution of the data set fluctuate. Also, spams received in the same time frame tend to have similar characteristics as far as the terms' frequencies are concerned. Consequently, we designed an update component that recalculates the words the appear most often in spams or hams and changes the list of features accordingly.

The last part of the application evaluates the results and informs the user and the system about the improvements needed. The efficacy of a classifier may be measured by its false positive rate (fpr) and false negative rate (fnr), defined as the proportion of spam (respectively ham) messages that it misclassifies. A classifier with lower fpr and fnr than another is superior. Whether a classifier with a lower fpr and higher fnr is superior or inferior depends on the user's sensitivity to each kind of error. A set of measures – including accuracy, weighted accuracy, total cost ratio, F-measure [11], and utility – attempt to quantify this sensitivity and to use this quantification to combine fpr and fnr, along with the corpus ham-to-spam ratio, into a one-dimensional measure.

IV. SPAM DATA COLLECTION AND FEATURE EXTRACTION

A more difficult problem is that of obtaining shareable corpora of non-spam email, which often contain personal details that people want to keep private. Two general approaches have been taken as presented in [6]: anonymizing the message by reducing it to word frequencies and performing feature selection upon the set, or using messages from websites and public mailing lists as proxies for personal email.

Each email is analyzed by its content type (multipart, text/plain, text/html, image, video, application etc) and several features are extracted. We extracted 57 numerical attributes and the target concept (spam/no spam) as nominal. 50 attributes are numerical, real, representing several words frequencies and the others are integers that count different message properties. We combined attributes extracted by other filters that use Naïve Bayes (words frequency [7] and their probability to appear in a spam or legitimate email [12]) with new ones extracted directly from message properties or from the message body as shown in Fig 4.

In order to pass through the simple filtering, spammers began to employ content obscuring techniques such as inserting false punctuation (i.e. "v.ia.g.ra", "100% Mo|ney Back Guaran|tee"), using bogus HTML tags and adding HTML comments in the middle of words (i.e. "Our pro<br2sd9/>duct is reco<br2sd9/> mmen<br2sd9 />ded and made from natu<br2sd9/>ral ingre<br2sd9 />dients". Because of the text distortion, spam filtering diverges significantly from most text classification and information retrieval problems, where authors are not deliberately trying to obfuscate content and defy indexing. The next step in processing the email is parsing the text from the body. We removed all html tags and encoded text, counting another two attributes: the number of sources and links that appear in the message. This way, if a key word is displayed intercalated with tags, in order to avoid spam filters, it will be discovered.

The other numerical attributes represent the frequency of the words that appear most often in either spam or legitimate email. For this process, every word is transformed using the Porter Stemming Algorithm [13], which removes the common morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval Systems. Ignoring the issue of precisely where the words originate, a document is represented by a vector of words, or terms. Terms with a common stem will usually have similar meanings, for example: connect, connected, connecting, connection, connections. Frequently, the performance of an

Extracted Features			
	integer (7)		***real (50)***
from	message properties	text properties	parsed text from the body
represent	number of recipients	message size	frequency of the top 50 words most used in spam or in legitimate email
	number of replies	number of sources	
	number of attachments	number of links	
	subject length		

Figure 4. Extracted features

Information Retrieval system will be improved if terms group such as they are conflated into a single term. This may be done by removing various suffixes -ED, -ING, -ION, IONS to leave the single term (connect). In addition, the suffix stripping process will reduce the total number of terms in the system, hence it will diminish the size and complexity of the data in the system, which is always advantageous. We also use a dictionary of the most common preposition, pronouns and adverbs used in English in order to ignore counting their frequency.

Before a word is stemmed, we removed all other characters beside letters. This way we avoid the case where a word is intentionally misspelled using numbers or signs (eg. "|" instead of "I", "3" instead of "e", "0" instead of "o"). If the words appear repeatedly in the same format, they will have the same normalized form.

Each processed and stemmed word's frequency in both spam and legitimate email is memorized. Only the top 50 words are used in the classification. The number of words used might be adjusted depending on the data set size and the message length. For a short message fewer words might be used as long as the difference between spam and legitimate message is still visible. In practice there are used 15 to 60 attributes regarding word frequency. We chose the top 50 words that appear more frequently in spam but not in legitimate email or vice-versa.

For this process, the attribute list is modified repeatedly in order to keep it up-to-date. Depending on the system settings, this step can occur after the analysis of either each email or a certain number of emails. Updating the list of attributes after each classified email is time consuming. It is also unlikely for the information from a single email to significantly change the general word frequencies. Still, if the accuracy is more important than the delay, this is the recommended solution. A middle solution, based on the similarity of the messages received in the same time frame, would be to run the update process after a certain number of emails or a certain period of time. In practice this depends on the frequency of the received emails.

For the attribute update process we used two hash tables: one mapping each email by the words that appear in it and their frequencies (the features table) and another one counting for each word the number of spams and hams it was used in (the words table). After an email is analyzed, each stemmed word is registered in the features table. If that term already exists in the words table, we increment the number of occurences for spam or ham depending on the email's class; otherwise we add a new record. In the update process, we recalculate the top 50 most frequently words and if they differ from the previous list we have to rebuild the data set. This involves going through the data set in order to remove old attributes and add new ones.

V. CLASSIFICATION MODULE

In the classification process the positive class represents the legitimate email while the negative class is represented by spam. A problem of real filtering that needs to be analyzed is the asymmetry in error costs. Judging a legitimate email to be spam (a false negative error) is usually far worse than judging a spam email to be legitimate (a false positive error). A false positive simply causes a slight irritation, i.e., the user sees an undesirable message. Conversely, a false negative can be critical. If spam is deleted permanently from an email server, a false negative can be very expensive since it means a (possibly important) message has been discarded without a trace. If spam is moved to a low-priority email folder for later human scanning, or if the address is only used to receive low priority email, false negatives may be much more tolerable.

Ken Schneider, CTO of the email filtering company Bright-Mail argues in [14] that filtering even a small amount of legitimate email defeats the purpose of filtering because it forces the user to start reviewing the spam folder for missed messages. Even a single missed important message may cause a user to reconsider the value of spam filtering.

Like most text classification domains, spam presents the problem of a skewed class distribution, i.e., the proportion of spam to legitimate email is uneven. The effect of class distribution and training-set size on classifier performance, analyzed in [10], gives support to the notion that there may be a "best" marginal class distribution for a learning task. It is suggested that a progressive sampling technique may be useful in locating the class distribution that yields the best, or nearly best, classifier performance. For any fixed class distribution, increasing the size of the training set always leads to improved classifier performance. Also by carefully selecting the class distribution, one can sometimes achieve improved performance while using fewer training examples. We performed several experiments in order to identify the most appropriate values for the data set size and class distribution.

The classification module resamples the available data set to the optimal size and distribution and then classifies the instances using the kNN algorithm. This is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to

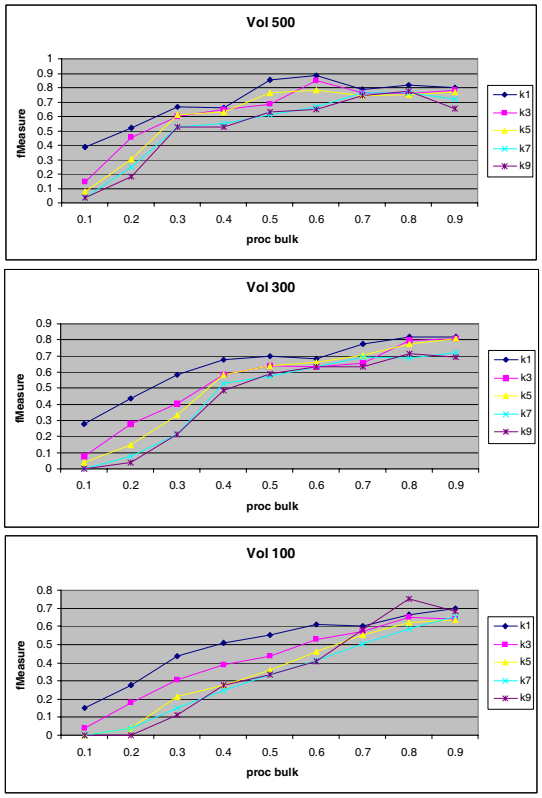


Figure 5. Identification of the most appropriate k value

the class of its nearest neighbor.

The main reason for selecting this algorithm was that it does not require any train step, so the updating of the most frequently words process is fast. Another reason was that it allows to implement an adaptive threshold, so a message would be classified as legitimate even if the probability to be spam is greater than 50%. This approach is recommended when the filter sensitivity is essential.

VI. EXPERIMENTAL WORK

In this section we present the experiments run to find the most appropriate values for k, data set size and legitimate email distribution. Also we made several

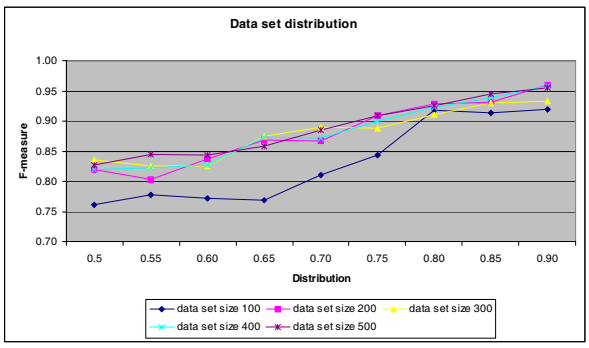


Figure 6. F-measure as a function of data set distribution and size, after a 10 fold cross validation with kNN

Data set size	Legitimate mail distribution	Correct classified rate	F measure	TP rate	TN rate
100	0.90	0.86	0.92	0.90	0.50
200	0.85	0.89	0.93	0.91	0.77
200	0.90	0.93	0.96	0.95	0.75
300	0.85	0.89	0.93	0.90	0.80
300	0.90	0.89	0.93	0.90	0.73
400	0.85	0.90	0.94	0.91	0.85
400	0.90	0.93	0.96	0.94	0.83
500	0.85	0.91	0.94	0.92	0.85
500	0.90	0.92	0.95	0.93	0.84

Figure 7. The best results for the combination of data set size, and positive class distribution, after a 10 fold cross validation with kNN

evaluations of the filter performance on static and dynamic data sets. The evaluation was performed on a 1000 instances data set, from which the necessary amount was selected. We used a part of the messages in the TREC 2005 Public Corpus [15] for our experiments. We processed 500 spam and 500 legitimate emails.

As performance metrics we used the true positive rate (TPR) and F-measure value. TPR, also known as sensitivity, or recall, measures the proportion of actual positives which are correctly identified as such form the total number of positives instances. The negative class is the true negative rate (TNR), also called specificity, and is computed as the proportion of negative examples correctly identified from the total number of negative instances. The positive predictive value (PPV), named also precision is the number of true positives divided by the total number of elements labeled as belonging to the positive class. F-measure is an assessment of a test's accuracy that considers the harmonic mean of precision and recall.

The first experiment was made to determine the most appropriate value for k in the kNN algorithm. The percentage of legitimate bulk has benn varied between 0.1-0.9 for each volume from 100 to 500 emails and in general the curved for k=1 had the best fit, as shown in Fig. 5.

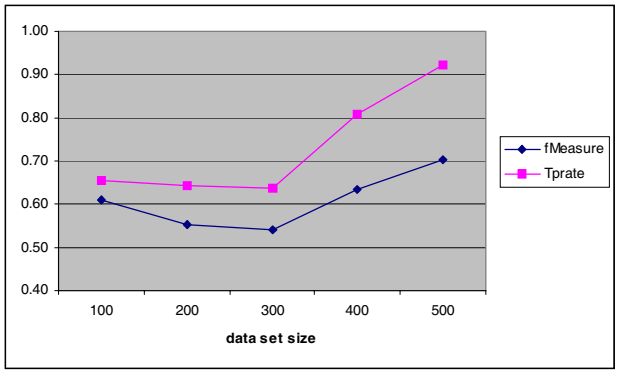


Figure. 8 Simple update component performance

Size	F-measure		Tprate	
	after resample	*without resample*	*after resample*	*without resample*
100	0.72	0.72	0.83	0.74
200	0.71	0.74	0.87	0.80
300	0.69	0.71	0.83	0.81
400	0.72	0.72	0.90	0.88
500	0.70	0.74	0.91	0.93

Figure 9. Filter performance with update, feedback and resample process

Next evaluation was run to identify the optimal data set size and email distribution of the positive class. The approach was to vary the size of the data set from 100 to 500 emails and the legitimate email percentage from 0.5 to 0.9. For each combination (data set size, positive class distribution), a 10 fold cross validation with kNN was performed. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. We used a stratified validation in order to maintain the same proportions of the two types of class labels in each fold. The values considered are the average results. Fig. 6 illustrates the evolution of the F-measure as a function of the data set distribution for each size and the table from Fig. 7 underlines the best results obtained.

For the best 3 values of the F-measure obtained we retested the filter performance, resampling the initial data set to the correlated values (size 200, 400, 500 and distribution 0.9). The F-measure and True Positive rate improved significantly as shown in the Fig. 7. The optimal values are for a size of 500 emails with a 0.9 legitimate email distribution. The F-measure for the initial filter was 0.87, TPR 0.82 and accuracy 0.87. The performance improved after the data set resampling to 0.98 for both F-measure and TPR and 0.97 for accuracy. In conclusion, F-measure raised with 13% and TPR with 19% and accuracy with 11%.

In the next step we tested the update component. We evaluated the filter on a test set of 100 emails and 0.5 distribution and a train set of variable size (100-500) and distribution (0.5-0.9). The results from Fig. 8 show that the update component performs worse in this situation than the process with no retrain. Still, the two results are not fully comparable since the first tests were performed combined with a 10 fold cross validation and the last one on a fixed test set. Another explanation would be that a misclassified message added to the train set propagates the error in classifying other messages. This suggested the improvement of the filter by adding a feedback option that will correct the errors. This option implies the user's action and judgement. For the feedback simulation, we remade the last evaluation and after each email being classified, we corrected the prediction before adding the

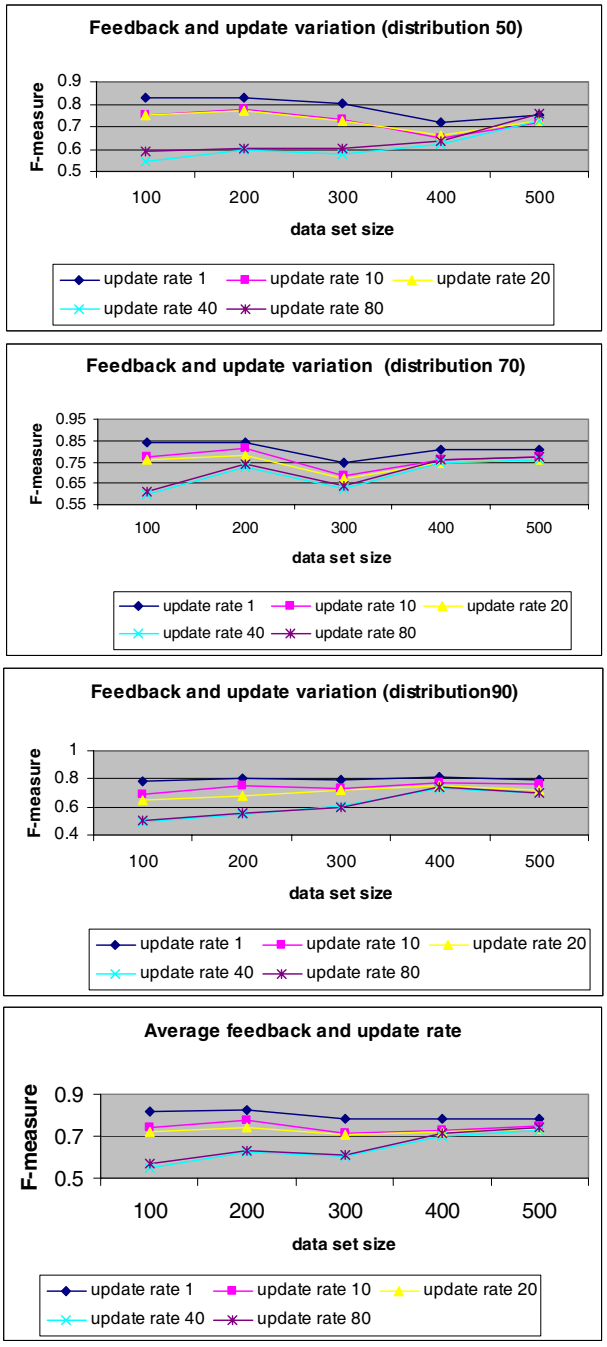
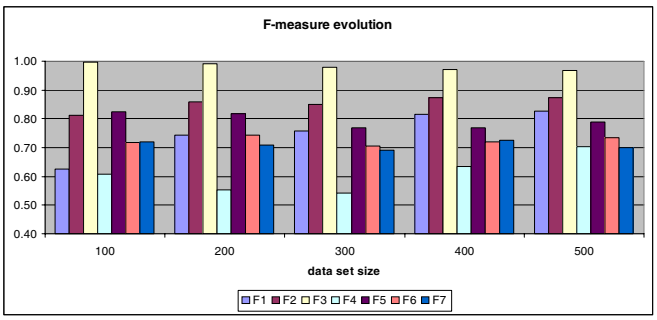


Figure 10. An analyze of different feedback rates

new instance to the train set. Results, as shown in Fig. 9, improved significantly.

Nevertheless, a feedback after each email is unlikely in a real time process since it depends on the user's actions. Therefore we tested several values(1, 10, 20, 40, 80) to find the most appropriate feedback and update rate for different data set size.



F1=F-measure for the initial test, on 100emails as test set

F2=F-measure from a 10fold CV of the data set

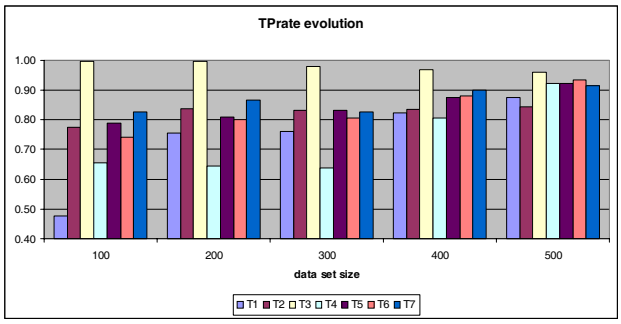
F3=F-measure after resampling the data set to 500emails size, 90% distribution

F4=F-measure after the update process

F5=F-measure for update and feedback with rate= 1 process

F6=F-measure for update and feedback with rate=20 process

F7=F-measure for update, feedback(rate 20) and resample (size 500, distribution 0.9) process



T1=TPrate for the initial test, on 100emails as test set

T2=TPrate from a 10fold CV of the data set

T3=TPrate after resampling the data set to 500emails size, 90% distribution

T4=TPrate after the update process

T5=TPrate for update and feedback with rate= 1 process

T6= TPrate for update and feedback with rate=20 process

T7= TPrate for update, feedback(rate 20) and resample (size 500, distribution 0.9) process

Figure 11. General performance evolution for (a) F-measure (b) TPrate

In Fig 10 are presented the results: as expected, the best performance is generated by rate=1 and then it deteriorates once the rate elevates. Still, in our case, rate=20 seems a proper value for feedback because it generates almost the same performance as rate=10 and is less costly.

For the last evaluation, we combined all suitable values obtained above(k=1, resample size=500, resample distribution=0.9, feedback/update rate=20) in order to simulate a real time spam filtering process. Each email is analyzed, classified and its features are memorized. After every series of 20 emails, the attributes are updated and the train set is rebuilt and resampled to 500 emails with a 0.9 distribution of the positive class. In Fig 11 the final results show an improvement for TPR, but not for the F-measure.

VII. CONCLUSIONS AND FURTHER WORK

In this paper we proposed a new approach for a spam detection filter. Messages are classified with the kNN algorithm based on a set of features extracted from the email's properties and content. The train set is resampled to the most appropriate size and positive class distribution determined by several experiments. The system performs a constant update of the data set and the list of most frequently words that appear in the messages. Also a feedback option regarding misclassified messages is implemented and recommended to be performed at a certain rate depending on the resources available.

We adapted the filter to each user's particularities. Further development would be to add the email's sending date as an attribute and change the kNN algorithm to set greater weights for the emails sent in the same time frame. Also performing a feature selection on the extracted attributes would be another direction for further possible study.

REFERENCES

[1] Junod J., "Servers to spam: drop dead, Computers and Security," Elsevier, vol. 16, 1997.

[2] Cormack, G. V. "Email Spam Filtering: A systematic review", vol. 1. 2008.

[3] Sahami M., Dumais S., Heckerman D., Horvitz E., "A bayesian approach to filtering junk e-mail." AAAI-98 Workshop on Learning for Text Categorization.

[4] Androutsopoulos I., Koutsias J., Chandrinos K.V., and Spyropoulos, C.D. 2000. "An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages", ACM SIGIR conference on Research and development in information retrieval. Athens, Greece, 2000.

[5] Ramachandran A., Feamster N. "Understanding the network-level behavior of spammers."(2006)

[6] Chih-Hung Wu, "Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks, Expert Systems with Applications". An International Journal, v.36 n.3, April, 2009

[7] Fumera, G., Pillai, I., Roli, F., "Spam Filtering Based On The Analysis Of Text Information Embedded Into Images", Journal of Machine Learning Research 7 (2006)

[8] Cormack G., Cheriton D., "Batch and Online Spam Filter Comparison" (2006)

[9] Gomez Hidalgo J. M.."Evaluating cost-sensitive unsolicited bulk email categorization.", ACM Symposium on Applied Computing, Madrid, ES, 2002.

[10] Weiss G. M., Provost F., "Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction", Journal of Artificial Intelligence Research 19 (2003)

[11] Hripcsak G., Rothschild S., "Agreement, the F-Measure, and Reliability in Information Retrieval" (2005)

[12] T. Fawcett, "In vivo Spam Filtering: A Challenge Problem for Data Mining". SigKDD Explorations, Vol.5, No.2. Dec. 2003.

[13] Porter, M.F., "An algorithm for suffix stripping".(1980)

[14] Schneider K.. "Fighting spam in real time." MIT Spam Conference, Jan 2003.

[15] Cormack, G. V., and Lynam, T. R. "Overview of the TREC 2005 Spam Evaluation Track". In Fourteenth Text REtrieval Conference (TREC-2005)