Enhancing Scalability in Anomaly-based Email Spam Filtering

Carlos Laorden
DeustoTech Computing - S^{3}Lab, University of Deusto
Avenida de las Universidades 24, 48007
Bilbao, Spain
claorden@deusto.es

Xabier Ugarte-Pedrero
DeustoTech Computing - S^{3}Lab, University of Deusto
Avenida de las Universidades 24, 48007
Bilbao, Spain
xabier.ugarte@deusto.es

Igor Santos
DeustoTech Computing - S^{3}Lab, University of Deusto
Avenida de las Universidades 24, 48007
Bilbao, Spain
isantos@deusto.es

Borja Sanz
DeustoTech Computing - S^{3}Lab, University of Deusto
Avenida de las Universidades 24, 48007
Bilbao, Spain
borja.sanz@deusto.es

Pablo G. Bringas
DeustoTech Computing - S^{3}Lab, University of Deusto
Avenida de las Universidades 24, 48007
Bilbao, Spain
pablo.garcia.bringas@deusto.es

ABSTRACT

Spam has become an important problem for computer security because it is a channel for the spreading of threats such as computer viruses, worms and phishing. Currently, more than 85% of received emails are spam. Historical approaches to combat these messages, including simple techniques such as sender blacklisting or the use of email signatures, are no longer completely reliable. Many solutions utilise machine-learning approaches trained using statistical representations of the terms that usually appear in the emails. However, these methods require a time-consuming training step with labelled data. Dealing with the situation where the availability of labelled training instances is limited slows down the progress of filtering systems and offers advantages to spammers. In a previous work, we presented the first spam filtering method based on anomaly detection that reduces the necessity of labelling spam messages and only employs the representation of legitimate emails. We showed that this method achieved high accuracy rates detecting spam while maintaining a low false positive rate and reducing the effort produced by labelling spam. In this paper, we enhance that system applying a data reduction algorithm to the labelled dataset, finding similarities among legitimate emails and grouping them to form consistent clusters that reduce the amount of needed comparisons. We show that this improvement reduces drastically the processing time, while maintaining detection and false positive rates stable.

CEAS '11 September 1-2, 2011, Perth, Western Australia, Australia

Categories and Subject Descriptors

K.4 [**Computers and Society**]: Electronic Commerce—*Security*

General Terms

Security

Keywords

email spam, anomaly detection, dataset clustering, computer security

1. INTRODUCTION

Electronic mail (email) is a powerful communication channel. Nevertheless, as happens with all useful media, it is prone to misuse. Flooding inboxes with annoying and time-consuming messages, more than 85% of received emails are spam[1]. Besides, bulk email not only is very annoying to every-day email users, but also constitutes a major computer security problem that costs billions of dollars in productivity losses [4]. Moreover, it can be used as a medium for *phishing* (i.e., attacks that seek to acquire sensitive information from end-users) [10] and the spread of *malicious software* (e.g., computer viruses, Trojan horses, spyware and Internet worms) [4].

Several approaches have been proposed by the academic community to solve the spam problem [16, 6, 21, 7]. Among them, the termed as *statistical approaches* [23] use machine-learning techniques to classify emails. These approaches have proved their efficiency detecting spam and are the most extended techniques to fight it. In particular, the use of the Bayes' theorem is widely used by anti-spam filters (e.g., SpamAssasin [14], Bogofilter [15], and Spamprobe [5]).

These statistical approaches are usually supervised, i.e., they need a training set of previously labelled samples. These techniques perform better as more training instances are

[1] http://www.junk-o-meter.com/stats/index.php

available, which means that a significant amount of previous labelling work is needed to increase the accuracy of the models. This work includes a gathering phase in which as many emails as possible are collected. However, the availability of labelled training instances is limited, which slows down the progress of anti-spam systems.

In a previous work, we presented the first spam filtering method based on anomaly detection [20] that reduces the necessity of labelling spam messages and only employs the representation of legitimate emails. This approach calculates vectors composed of certain features and compares the samples against a set of vectors representing legitimate emails. If the compared sample is sufficiently different, then it is considered as spam. Although the results obtained were significant enough to validate our method, the number of comparisons needed to analyse each sample was considerably high and consequently, it presented a high processing overhead.

In consideration of this background, we propose here an enhancement of our previous method that applies partitional clustering to the dataset in order to reduce the number of vectors in the dataset used as normality. This improvement boosts scalability due to the reduction in the processing time.

Summarising, our main contributions are:

- We adapt a method for email dataset reduction based on the partitional clustering algorithm Quality Threshold (QT) clustering, and generate reduced datasets of different sizes.
- We empirically validate the reduction algorithm testing its accuracy results and comparing them to previous work.
- We prove that a unique synthetically generated sample for legitimate emails is representative enough to implement an anomaly detection system without compromising accuracy results.

The remainder of this paper is organised as follows. Section 2 details our anomaly-based method. Section 3 describes the experiments and presents results. Section 4 discusses the obtained results and their implications, and outlines avenues for future work.

2. METHOD DESCRIPTION

The method described in this paper is based on a previous work, in which we presented an anomaly-based spam filter [20]. We have improved its efficiency by designing a data reduction phase capable of boosting the scalability of the filtering system.

The first step consists in the representation of the emails comprising both datasets. Then we apply the clustering algorithm to them, to obtain a reduced version that conserves the original datasets' characteristics. Finally, the anomaly detection step is performed. Due to the sample reduction phase, the number of comparisons performed decreases and, thus, the comparison time required for the analysis of each sample is much lower.

2.1 Representation of emails

As other spam filtering systems, our approach attempts to accurately classify email massages into 2 main categories: spam or not spam (i.e., legitimate emails). To this end, we use the information found within the body and subject of an email message and discard every other piece of information (e.g., the sender or the time-stamp of the email). To represent messages, we start by removing stop-words [22], which are words devoid of content (e.g., 'a','the','is'). These words do not provide any semantic information and add noise to the model [18].

Afterwards, we represent the emails using an Information Retrieval (IR) model. An IR model can be defined as a 4-tuple $[\mathcal{E}, \mathcal{Q}, F, R(q_i, e_j)]$ [2] where $\mathcal{E}$, is a set of representations of emails; F is a framework for modelling emails, queries and their relationships, $\mathcal{Q}$ is a set of representations of user queries and, finally, $R(q_i, e_j)$ is a ranking function that associates a real number with a query q_i ($q_i \in \mathcal{Q}$) and an email representation e_j, so that ($e_j \in \mathcal{E}$).

As $\mathcal{E}$ is the set of text emails e, $\{e : \{t_1, t_2, ...t_n\}\}$, each comprising n terms $t_1, t_2, \ldots, t_n$, we define the weight $w_{i,j}$ as the number of times the term t_i appears in the email e_j. If $w_{i,j}$ is not present in e, $w_{i,j} = 0$. Therefore, an email e_j can be represented as the vector of weights $\vec{e_j} = (w_{1,j}, w_{2,j}, ...w_{n,j})$.

On the basis of this formalisation, we use the well known *Term Frequency – Inverse Document Frequency* (TF–IDF) [18] weighting schema, where the weight of the i^{th} term in the j^{th} document, is defined by $weight(i,j) = tf_{i,j} \cdot idf_i$, where *term frequency* is defined as $tf_{i,j} = (n_{i,j})/(\sum_k n_{k,j})$, $n_{i,j}$ is the number of times the term $t_{i,j}$ appears in a document d, and $\sum_k n_{k,j}$ is the total number of terms in the document d. The inverse term frequency is defined as $idf_i = (|\mathcal{D}|)/(|\mathcal{D} : t_i \in d|)$, where $|\mathcal{D}|$ is the total number of documents and $|\mathcal{D} : t_i \in d|$ is the number of documents containing the term t_i.

2.2 Data reduction

Dataset reduction is a step that has to be faced in very different problems when working with large datasets. In our previous work [20], the experiments were performed with a base of more than 2,000 (for the LingSpam dataset) and more than 4,000 (for the SpamAssassin dataset) legitimate emails, which means that every sample analysed had to be compared 2,000 or 4,000 times to classify it as spam or not. Therefore, we propose a data reduction algorithm based on partitional clustering.

Cluster analysis divides data into meaningful groups [13]. These techniques usually employ distance measures to compare instances in datasets to make groups with those which appear to be similar. We can identify several types of clustering, but the most common ones are hierarchical clustering and partitional clustering. The first approach generates clusters in a nested style, which means that the dataset is divided into a set of clusters which are subdivided into other clusters related hierarchically. In contrast, partional clustering techniques create a one-level (unnested) partitioning of the data points [13]. We are interested in this last technique to validate our initial hypothesis: it is possible to divide a big set of emails that represent normality (i.e., legitimate emails) into a reduced set of representations.

QT clustering algorithm was proposed by Heyer et al. [9] to extract useful information from large amounts of gene expression data. This clustering algorithm does not need to specify the number of clusters desired. Concretely, it uses a similarity threshold value to determine the maximum ra-

input : The original dataset $\mathcal{V}$, the distance threshold for each cluster *threshold*, and the minimum number of vectors in each cluster *minimumvectors*
output: The reduced dataset $\mathcal{R}$

```
// Calculate the distance from each vector (set
   of email features) to the rest of vectors in
   the dataset.
foreach {v_i | v_i ∈ V} do
    foreach {v_j | v_j ∈ V} do
        // If a vector v_j's distance to v_i is
           lower than the specified threshold,
           then v_j is added to the potential
           cluster A_i, associated to the v_i
           vector
        if distance(v_i,v_j) ≥ threshold then
            A_i.add(v_j)
// In each loop, select the potential cluster
   with the highest number of vectors
while ∃A_i ∈ A : |A_i| ≥ minimumvectors and
∀A_j ∈ A : |A_i| ≥ |A_j| and i ≠ j do
    // Add the centroid vector for the cluster
       to the result set
    R.add(centroid(A_i))
    // Discard potential clusters associated to
       vectors v_j ∈ A_i
    foreach {v_j | v_j ∈ A_i} do
        A.remove(A_j)
        V.remove(v_j)
    // Remove vectors v_j ∈ A_i from the clusters
       A_k remaining in A
    foreach {A_k | A_k ∈ A} do
        foreach {v_j | v_j ∈ A_k and v_j ∈ A_i} do
            A_k.remove(v_j)
// Add the remaining vectors to the final
   reduced dataset
foreach {v_j | v_j ∈ V} do
    R.add(v_j)
```

Figure 1: QT Clustering based dataset reduction algorithm.

dial distance of any cluster. This way, it generates a variable number of clusters that meet a quality threshold. Its main disadvantage is the high number of distance calculations needed. Nevertheless, in our case, this computational overhead is admissible because we only have to reduce the dataset once, (we employ an static representation of normality that only varies from platform to platform).

Our algorithm, shown in Figure 1, is based on the concepts proposed by Heyer et al. [9], but it is adapted to our data reduction problem and it is implemented iteratively, instead of recursively.

Formally, let $\mathcal{A} = \{\mathcal{A}_0, \mathcal{A}_1, ..., \mathcal{A}_n\}$ be the set of potential clusters. For each vector v_i in the dataset $\mathcal{V}$, there is a potential cluster $\mathcal{A}_i \in \mathcal{A}$. A potential cluster $\mathcal{A}_i$ is composed of the set of vectors at a distance respect to v_i not higher than the *threshold* previously specified.

Once the potential clusters are calculated, we select the cluster with the highest number of vectors as a final cluster. Then, we calculate its centroid, defined as $c = x_1 + x_2 + \cdots + x_k/k$ where $x_1, x_2, \cdots, x_k$ are points in the feature space. The resultant centroid is added to the final reduced dataset. Afterwards, each vector v_j present in the selected cluster $\mathcal{A}_i$ is removed from the original dataset $\mathcal{V}$ (as they will be represented by the previously calculated centroid). Moreover, the potential clusters $\mathcal{A}_j \in \mathcal{A}$ associated to each vector v_j previously removed are also discarded. When there are not more clusters available with a number of vectors higher than the parameter *minimumvectors*, the remaining vectors in $\mathcal{V}$ are added to the final reduced dataset and the algorithm finishes and returns the resulting reduced dataset. The final result is a dataset composed of one centroid representing each cluster and all the vectors that were not associated to any cluster by the QT clustering algorithm.

2.3 Anomaly Detection

The features described represent each email as a point in the feature space. Our anomaly detection system analyses points in the feature space and classifies emails based on their similarity. The analysis of an email consists of 2 different phases:

- Extraction of the features from the email.
- Measurement of the distance from the point representing the email to the points that symbolise normality (i.e., legitimate emails).

As a result, any point at a distance from normality that surpasses an established threshold is considered an anomaly and, thus, a spam message. In this study, we have considered 2 different distance measures:

- **Manhattan Distance.** This distance between two points v and u is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes:

$$d(x, y) = \sum_{i=0}^{n} |x_i - y_i| \quad (1)$$

  where x is the first point; y is the second point; and x_i and y_i are the ith components of the first and second point, respectively.

- **Euclidean Distance.** This distance is the length of the line segment connecting two points. It is calculated as:

$$d(x, y) = \sum_{i=0}^{n} \sqrt{x_i^2 - y_i^2} \quad (2)$$

  where x is the first point; y is the second point; and x_i and y_i are the ith components of the first and second point, respectively.

Since we have to compute this measure with a variable number of points representing legitimate messages, a combination metric is required in order to obtain a final distance value which considers every measure performed. To this end, we employ 3 simple rules:

- **Mean rule.** Select the mean distance value of the computed distances.
- **Max rule.** Select the highest distance value.
- **Min rule.** Select the lowest distance value.

In this way, when our method analyses an email, the final distance value is dependant on the distance measure and the combination rule selected.

3. EMPIRICAL VALIDATION

To evaluate the performance of our method we have conducted an experiment consisting of 2 phases: firstly, we reduce the set of vectors corresponding to the representation of the legitimate emails that represent normality and, secondly, we start the anomaly detection step to measure both accuracy and efficiency.

3.1 Experimental Configuration

We used the *Ling Spam*[2] and *SpamAssassin*[3] datasets.

The SpamAssassin public mail corpus is a selection of 1,897 spam messages and 4,150 legitimate emails.

Ling Spam consists of a mixture of both spam and legitimate messages retrieved from the *Linguistic list*, an email distribution list about *linguistics*. It comprises 2,893 different emails, of which 2,412 are legitimate emails obtained by downloading digests from the list and 481 are spam emails retrieved from one of the authors of the corpus (for a more detailed description of the corpus please refer to [1, 17]). From the 4 different datasets provided in this corpus, each of one with different pre-processing steps, we chose the *Bare* dataset, which has no pre-processing.

We performed for both datasets a *Stop Word Removal* [22] based on an external stop-word list[4] and removed any non alpha-numeric character.

Then, we used the *Vector Space Model* (VSM) [19], an algebraic approach for *Information Filtering* (IF), *Information Retrieval* (IR), indexing and ranking, to create the model. This model represents natural language documents in a mathematical manner through vectors in a multidimensional space.

We extracted the top 1,000 attributes using *Information Gain* [11], an algorithm that evaluates the relevance of an attribute by measuring the relevance with respect to the class:

$$IG(j) = \sum_{v_j \in R} \sum_{C_i} P(v_j, C_i) \cdot (P(v_j, C_i)/(P(v_j) \cdot P(C_i)) \quad (3)$$

where C_i is the i^{th} class, v_j is the value of the j^{th} interpretation, $P(v_j, C_i)$ is the probability that the j^{th} attribute has the value v_j in the class C_i, $P(v_j)$ is the probability that the j^{th} interpretation has the value v_j in the training data, and $P(C_i)$ is the probability of the training dataset belonging to the class C_i.

[2]http://nlp.cs.aueb.gr/software_and_datasets/lingspam_public.tar.gz
[3]http://spamassassin.apache.org/publiccorpus/
[4]http://www.webconfs.com/stop-words.php

In order to evaluate the performance of the predictors, *k-fold cross validation* [12] is commonly used in machine-learning experiments [3]. For SpamAssasin, we performed a 5-fold cross-validation to divide the dataset composed of legitimate emails (the normal behaviour) into 5 different divisions of 3,320 emails for representing normality and 830 for measuring deviations within legitimate emails. In this way, each fold is composed of 3,320 legitimate emails that will be used as representation of normality and 2,726 testing emails, from which 830 are legitimate emails and 1,896 are spam.

With regards to Ling Spam dataset, we also performed a 5-fold cross-validation [12] forming 3 different divisions of 1,930 emails and two divisions of 1,929 emails for representing normality and other 3 divisions of 482 emails and 2 of 483 for measuring deviations within legitimate email. In this way, each fold is composed of 1,930 or 1,929 legitimate emails that will be used as representation of normality and 963 or 962 testing emails, from which 483 or 482 were legitimate emails and 480 were spam. The number of legitimate emails varied in the two last folds because the number of legitimate emails is not divisible by 5.

To test the dataset reduction algorithm proposed, 4 experimental configurations were selected for each distance measure. The threshold parameter values for our QT clustering based algorithm were selected by empirical observation. Table 1 shows the reductions obtained in the process. Reduction ratio varies from 13.21% for Euclidean distance and threshold 1.50 to 99.94% for both Euclidean and Manhattan distance and an infinite threshold (in practice, this threshold is set to the maximum value allowed for a 64-bit double variable). The result obtained for the infinity threshold is a unique centroid of the whole dataset that represents the arithmetic mean vector, or a single representation of normality. In this case, selection rules do not influence the final result because it is only performed one single comparison for each sample.

3.2 Efficiency Results

During our experimental evaluation, we measured the times employed in both data reduction and anomaly detection:

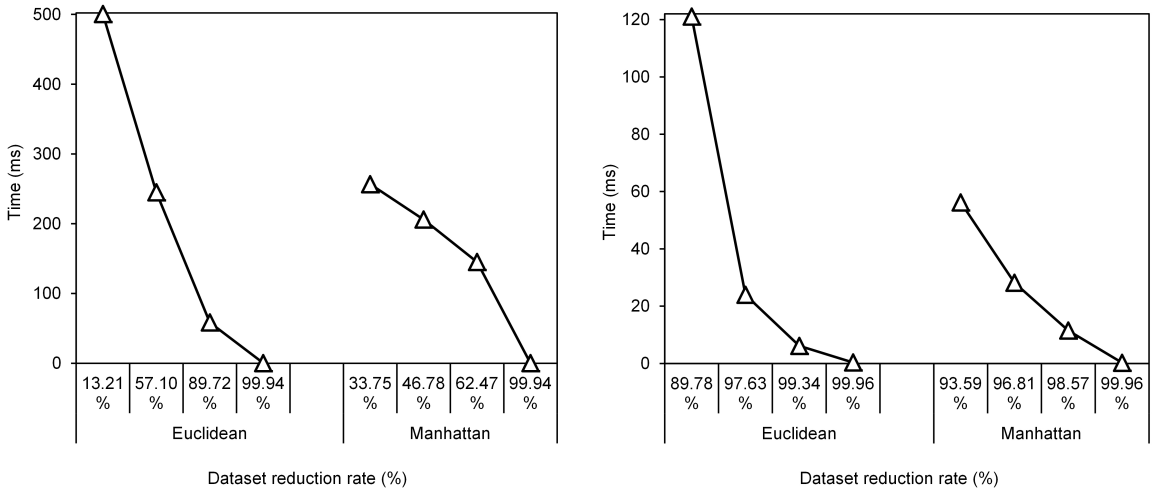
- **Data reduction.** In this phase, we reduced the original datasets for each fold. In this way, we used 8 different configurations to reduce each different dataset: Euclidean distance (1.50, 1.75, 2.00 and ∞) and Manhattan distance (1.50, 1.75, 2.00 and ∞).

  The average processing time consumed to reduce the datasets with each configuration is 1,107 seconds for LingSpam and 3,302 seconds for SpamAssassin when using Euclidean distance and 751 seconds for LingSpam and 2,179 seconds for SpamAssassin when using Manhattan distance. Note that this process, in spite of being very time consuming, is executed only once and does not interfere with the performance of the system.

- **Sample comparison.** In this phase, for each experimental configuration employed in the data reduction stage, the samples under test were compared against the reduced dataset. The number of comparisons depends exclusively on the number of vectors present in the resulting datasets, so it is straightforward that the

Table 1: Number of vectors conforming the reduced datasets for the different reduction parameters.

LingSpam							
Distance measure	**Quality threshold**	**% Average reduction**	**Vectors per fold**				
			1	2	3	4	5
Euclidean	1.50	13.21%	1,647	1,646	1,674	1,688	1,718
	1.75	57.10%	800	802	817	848	871
	2.00	89.72%	184	184	191	212	220
	∞	99.94%	1	1	1	1	1
Manhattan	1.50	33.75%	1,318	1,322	1,296	1,223	1,232
	1.75	46.78%	1,079	1,047	1,051	979	978
	2.00	62.47%	769	749	750	673	679
	∞	99.94%	1	1	1	1	1
SpamAssassin							
Distance measure	**Quality threshold**	**% Average reduction**	**Vectors per fold**				
			1	2	3	4	5
Euclidean	1.50	89.78%	302	342	431	297	324
	1.75	97.63%	66	79	102	66	79
	2.00	99.34%	16	18	33	20	21
	∞	99.96%	1	1	1	1	1
Manhattan	1.50	93.59%	119	230	251	221	242
	1.75	96.81%	50	117	132	109	121
	2.00	98.57%	17	53	60	52	54
	∞	99.96%	1	1	1	1	1



(a) Time required by the comparison phase for each reduced dataset of LingSpam

(b) Time required by the comparison phase for each reduced dataset of SpamAssassin

Figure 2: The X axis represents the resulting reduction rate for each dataset once the clustering step was applied. The bigger the reduction rate, the lower the number of vectors utilised. The Y axis represents the average comparison time for each executable file, expressed in milliseconds.

time employed in this step is inversely proportional to the threshold value used in the clustering algorithm. Figure 2 shows the average time employed by the comparison step for each testing sample. It can be noticed that the time required for comparison is lower when fewer vectors are utilised. For euclidean distance the average comparison time varies from 494.53 ms for a 1.50 clustering threshold value, to 0.46 ms for an ∞ threshold (comparison against a single vector representation) with LingSpam and from 121.07 ms for a 1.50 threshold to 0.35 for an ∞ threshold with SpamAssassin. In the case of manhattan distance, times are lower due to the simplicity of the calculations needed, varying from 257.55 ms and 56.22 for a 1.50 clustering threshold value to 0.30 ms and 0.24 for ∞ threshold with LingSpam and SpamAssassin respectively. The reduced comparison times offered by SpamAssassin when comparing it with LingSpam are due to the increased reduction suffered by the first dataset (as shown in Table 1).

3.3 Efficacy Results

Hereafter, we obtained the representation of the emails from both datasets (LingSpam and SpamAssassin), reduced

Table 2: Results for the different reduced datasets of LingSpam, combination rules and distance measures.

LingSpam						
Dataset		**Selection rule**	**Threshold**	**Prec.**	**Rec.**	**AUC**
Euclidean	**Prev. work**	Mean	2.59319	0.92821	0.91583	0.95017
		Max	4.15651	0.85953	0.79292	0.85725
		Min	1.93707	0.87510	0.93125	0.93944
	1.50	Mean	2.61855	0.93012	0.90958	0.94963
		Max	4.15651	0.85953	0.79292	0.85725
		Min	2.01180	0.93285	0.90292	0.95180
	1.75	Mean	2.72416	0.93929	0.90250	0.95162
		Max	4.15651	0.85953	0.79292	0.85725
		Min	2.05017	0.94598	0.88292	0.95563
	2.00	Mean	2.91093	0.94501	0.88792	0.95028
		Max	4.15647	0.85953	0.79292	0.85725
		Min	2.08618	0.95385	0.85250	0.95542
	∞	Mean	2.11057	0.95802	0.83667	0.95461
		Max	2.11057	0.95802	0.83667	0.95461
		Min	2.11057	0.95802	0.83667	0.95461
Manhattan	**Prev. work**	Mean	4.04255	0.79183	0.73542	0.83851
		Max	5.81974	0.76655	0.74292	0.83054
		Min	2.59853	0.56694	0.91042	0.77710
	1.50	Mean	3.97401	0.84907	0.73833	0.86281
		Max	5.81974	0.76655	0.74292	0.83054
		Min	2.91339	0.76078	0.81625	0.88033
	1.75	Mean	4.08539	0.87605	0.73917	0.87027
		Max	5.81974	0.76655	0.74292	0.83054
		Min	3.05884	0.79812	0.81542	0.89099
	2.00	Mean	4.22296	0.90349	0.74500	0.88178
		Max	5.81974	0.76655	0.74292	0.83054
		Min	3.20269	0.93333	0.72333	0.89595
	∞	Mean	3.58608	0.87308	0.75667	0.87198
		Max	3.58608	0.87308	0.75667	0.87198
		Min	3.58608	0.87308	0.75667	0.87198

the dataset using the 2 different distance measures and 4 different threshold values (resulting into 16 different reduced datasets), and employed the same 2 different measures and the 3 combination rules described in Section 2.3 to test the datasets and obtain a final measure of deviation for each testing sample.

For each measure and combination rule, we established 10 different thresholds to determine whether an email is spam or not, and selected the one which conducted to the best results in each case.

We evaluated the accuracy by measuring *precision*, *recall*, and *Area Under the ROC Curve* (AUC). We measured the precision of the spam identification as the number of correctly classified spam emails divided by the number of correctly classified spam emails and the number of legitimate emails misclassified as spam, $S_P = N_{s\to s}/(N_{s\to s} + N_{l\to s})$, where $N_{s\to s}$ is the number of correctly classified spam and $N_{l\to s}$ is the number of legitimate emails misclassified as spam.

Additionally, we measured the recall of the spam email messages, which is the number of correctly classified spam emails divided by the number of correctly classified spam emails and the number of spam emails misclassified as legitimate, $S_R = N_{s\to s}/(N_{s\to s} + N_{s\to l})$, where $N_{s\to s}$ is the number of correctly classified spam and $N_{s\to l}$ is the number of spam emails misclassified as legitimate.

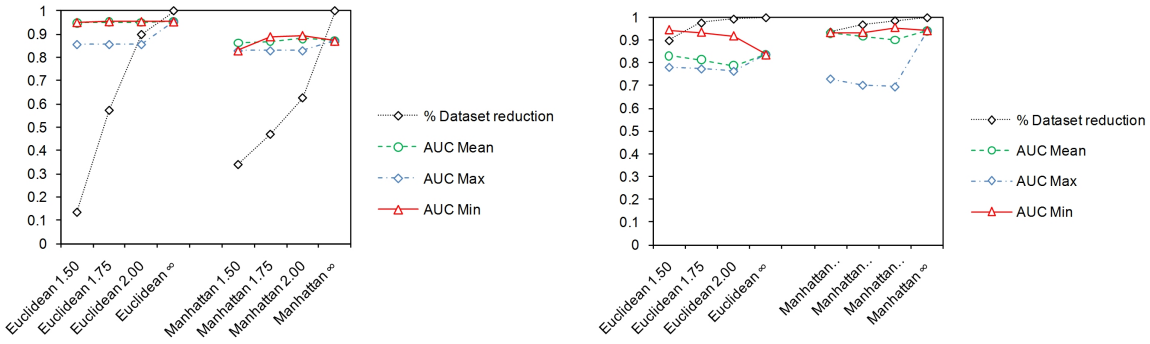
Finally, the AUC is defined as the area under the curve formed by the union of the points representing False Positive Rate (FPR) and True Positive Rate (TPR) for each possible threshold in a plot where the X axis represents the FPR and the Y axis represents the TPR. To calculate the AUC we used the points corresponding to the 10 thresholds selected. The lowest and the highest thresholds were selected in such a way that they produced a 0% of False Negative Rate (FNR) and a 0% of FPR respectively. The rest of thresholds were selected by equally dividing the range between the first and the last threshold. The area under the curve formed by these points was calculated dividing it into 9 trapezoidal subareas and computing them independently:

$$AUC = \sum_{i=0}^{i=9} (x_{i+1} - x_i) \cdot y_i + \frac{(x_{i+1} - x_i) \cdot (y_{i+1} - y_i)}{2}$$

Tables 2 and 3 show the obtained results. To simplify the results presented in Tables 2 and 3, we only show the performance associated to the best threshold for each configuration. Our anomaly-based spam filtering system is able to correctly detect more than 95 % of junk mails while maintaining the rate of misclassified legitimate emails lower than 5 % for the best configuration tested with SpamAssassin (Manhattan distance, 2.00 threshold and Min rule) and an

Table 3: Results for the different reduced datasets of SpamAssassin, combination rules and distance measures.

SpamAssassin						
Dataset		**Selection rule**	**Threshold**	**Prec.**	**Rec.**	**AUC**
Euclidean	**Prev. work**	Mean	2.13512	0.76144	0.97774	0.83574
		Max	3.83970	0.72990	0.97658	0.80730
		Min	1.39962	0.92103	0.93998	0.93987
	1.50	Mean	2.27873	0.77587	0.97278	0.83044
		Max	3.66095	0.72592	0.98481	0.78032
		Min	1.47257	0.95677	0.87553	0.94392
	1.75	Mean	2.43757	0.76615	0.96973	0.81427
		Max	3.64738	0.72674	0.98217	0.77337
		Min	1.50019	0.95161	0.86297	0.93374
	2.00	Mean	2.54008	0.73190	0.97795	0.78741
		Max	3.58818	0.71212	0.98112	0.76317
		Min	1.51793	0.92955	0.85876	0.91770
	∞	Mean	1.55007	0.76339	0.98186	0.83639
		Max	1.55007	0.76339	0.98186	0.83639
		Min	1.55007	0.76339	0.98186	0.83639
Manhattan	**Prev. work**	Mean	2.44136	0.91033	0.92848	0.93379
		Max	5.29903	0.69589	0.99958	0.73404
		Min	1.37525	0.91527	0.97764	0.96504
	1.50	Mean	3.01706	0.91716	0.91793	0.92964
		Max	5.29349	0.69558	1.00000	0.72957
		Min	2.07288	0.90819	0.93597	0.93198
	1.75	Mean	3.14866	0.89718	0.91857	0.91760
		Max	5.29349	0.69552	1.00000	0.70210
		Min	2.07288	0.94005	0.89483	0.93266
	2.00	Mean	3.52854	0.83624	0.95179	0.90021
		Max	4.97643	0.69558	1.00000	0.69474
		Min	2.33412	0.91584	0.95274	0.95378
	∞	Mean	2.37407	0.91779	0.93270	0.93985
		Max	2.37407	0.91779	0.93270	0.93985
		Min	2.37407	0.91779	0.93270	0.93985



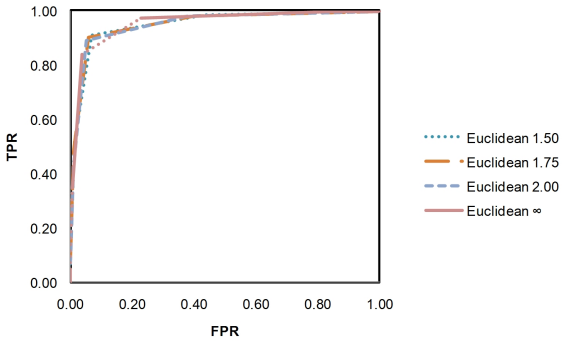
(a) Dataset reduction rate and the accuracy achieved with each reduced dataset of LingSpam.

(b) Dataset reduction rate and the accuracy achieved with each reduced dataset of SpamAssassin.

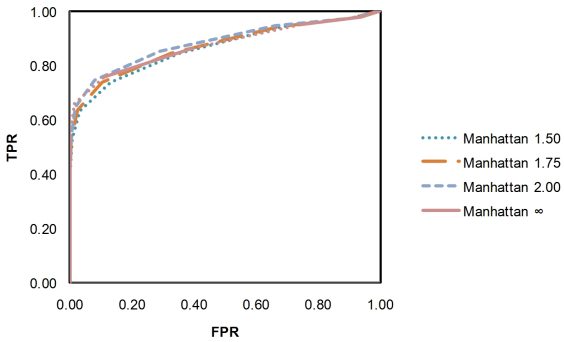
Figure 3: The continuous line represents the increasing reduction rate (the higher the rate, the lower the number of samples in the reduced dataset), while the dotted lines represent the area under the ROC curve (AUC) obtained with each reduced dataset.

improvable 10 % rate with LingSpam (Euclidean distance, 1.50 of threshold and Min rule). As it can be observed, min combination rule achieved the best results for all configurations.

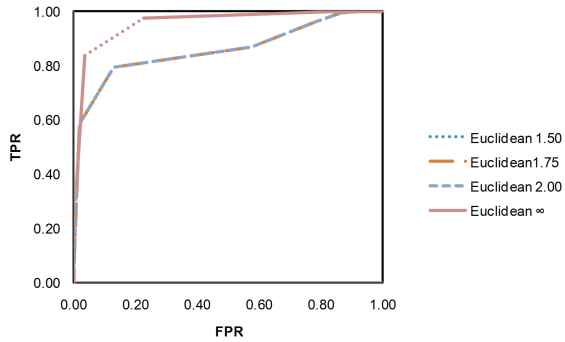
Figures 4 and 5 show 6 different plots for each different distance measure and selection rule. In each plot we can observe 4 ROC curves corresponding to the 4 different reduced datasets. We can observe that in some of the cases, the ROC curve shows better results when the threshold employed for reduction is ∞ (and thus, the number of vectors to compare with, is only 1). Figures 3(a) and 3(b) represent the evolution for the Max selector. Concretely, as the number of vectors is reduced, the system looses accuracy for SpamAssassin and maintains it for LingSpam. Nevertheless,



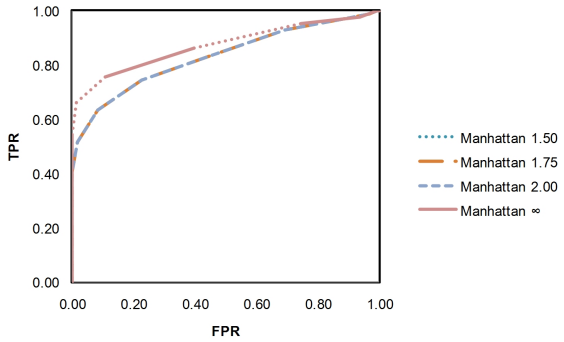
(a) ROC curve for the Euclidean distance and Mean selector.



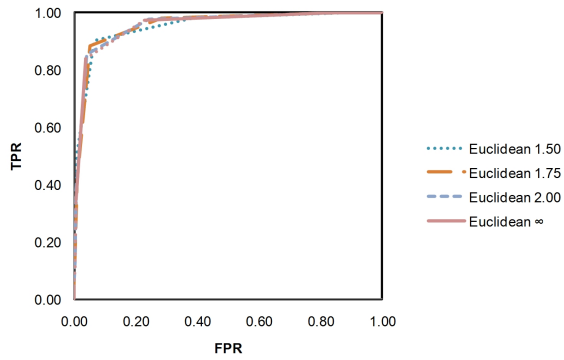
(b) ROC curve for the Manhattan distance and Mean selector.



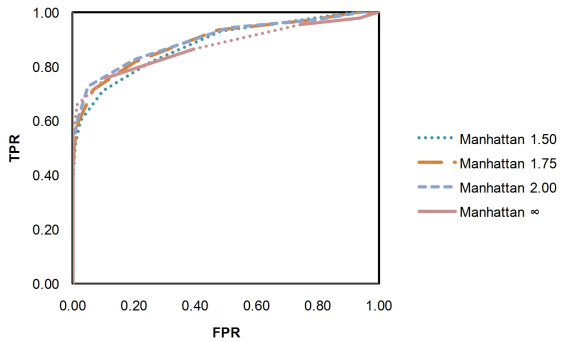
(c) ROC curve for the Euclidean distance and Max selector.



(d) ROC curve for the Manhattan distance and Max selector.



(e) ROC curve for the Euclidean distance and Min selector.



(f) ROC curve for the Manhattan distance and Min selector.

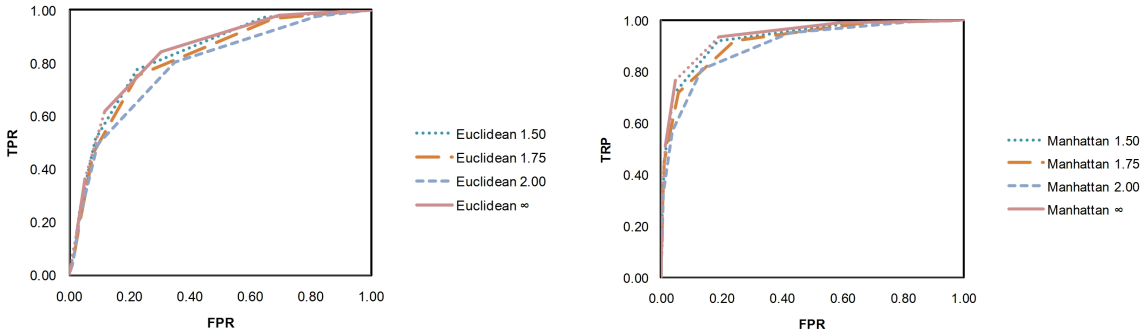
Figure 4: ROC curves for the different experimental configurations applied to LingSpam. Each figure shows 4 ROC curves corresponding to the different reduced datasets.

when the samples are compared against the mean vector, the results obtained improve and are even better than the ones achieved for the not reduced dataset (Euclidean and Manhattan distances with Max selector, shown in Figures 4(c), 5(c), 4(d), 5(d)).

This behaviour is more noticeable for Max selector, owing to the fact that this selector is more sensitive to groups of vectors distant from the normality representation and can affect in a negative way as it alters the distance value obtained. In contrast, the Min selector achieved the best results in all cases. In particular, the results obtained for min selector with an ∞ threshold are sounder than the ones obtained without reduction in the case of the LingSpam dataset but are worse for the SpamAssassin dataset.

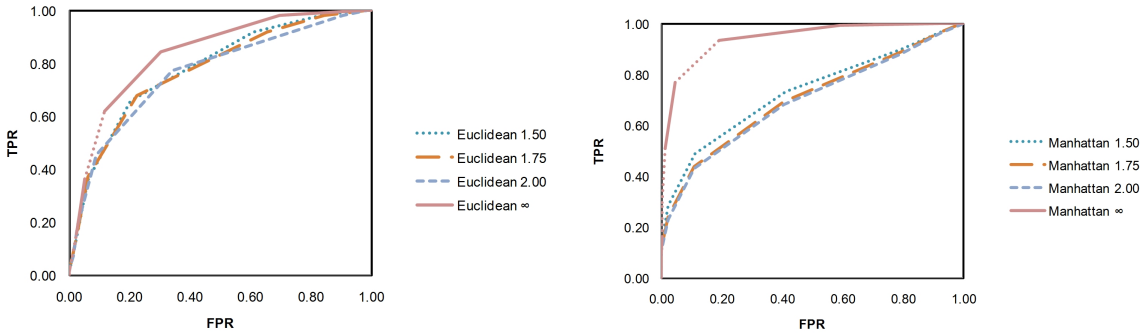
4. DISCUSSION AND CONCLUSIONS

This method improves a previous work where we presented the first spam filtering method based on anomaly detection [20] that reduces the necessity of labelling spam messages and only employs the representation of legitimate emails. In this paper, we enhance that system applying



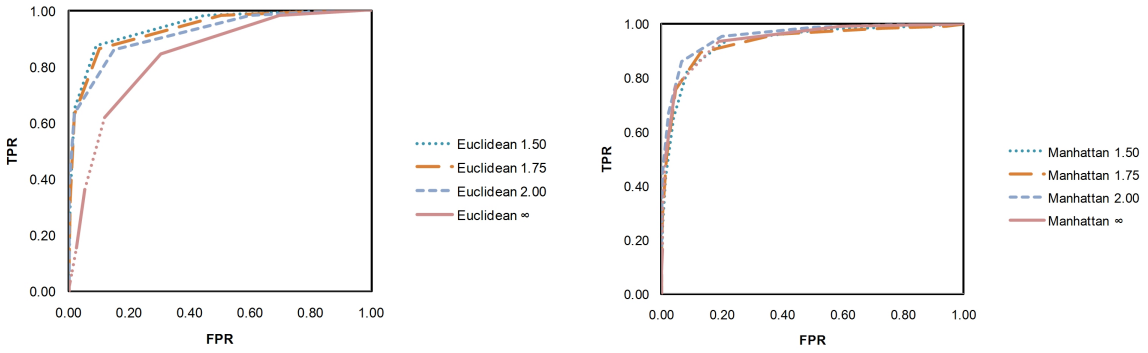
(a) ROC curve for the Euclidean distance and Mean selector.

(b) ROC curve for the Manhattan distance and Mean selector.



(c) ROC curve for the Euclidean distance and Max selector.

(d) ROC curve for the Manhattan distance and Max selector.



(e) ROC curve for the Euclidean distance and Min selector.

(f) ROC curve for the Manhattan distance and Min selector.

Figure 5: ROC curves for the different experimental configurations applied to SpamAssassin. Each figure shows 4 ROC curves corresponding to the different reduced datasets.

a data reduction algorithm that boosts scalability in the anomaly detection process, enabling a much more efficient comparison of samples. We show that this improvement reduces drastically the processing time, while maintaining detection and false positive rates stable. As opposite to other approaches, anomaly detection systems do not need previously labelled data of both classes (i.e., spam and legitimate), as they measure the deviation of email samples to normality (legitimate emails). In this way, this approach reduces significantly the labelling efforts required by machine-learning methods.

Although anomaly detection systems tend to produce high false positive rates, our experimental results show very low values in every case. Furthermore, accuracy is not affected by the dataset reduction process. It can be observed that the *AUC* does not vary significantly as the number of vectors in the dataset decreases. Even when a single centroid vector is used, results are still sound, or, in some cases, even better than the ones obtained with no reduction. This fact brings us to the conclusion that it is possible to determine a single representation for legitimate emails, and that this single representation is sufficiently accurate to classify as spam any sample whose representation deviates from our model.

Nevertheless, there are some limitations that should be

taken into account in further work. First, the Quality Thresholds and the thresholds of each selection rule have been selected through empirical observation. An extensive analysis is mandatory in order to detect possible optimisations. An automated process could select the best threshold combinations to improve the results of our filtering system. Furthermore, the difference in the results between LingSpam and SpamAssassin datasets will be subject to further study to provide a more comprehensive understanding of the behaviour of our proposed method with datasets of different nature.

Secondly, this study assumes that spam emails represent the anomalous messages. According to nowadays junk email traffic volume and to the characteristics of this type of messages this assumption may be incorrect. Further work will focus on this topic, comparing both approaches, spam as anomaly and legitimate emails as anomaly.

Finally, the datasets we employed should be extended to more current ones in order to test the method against bigger collections in size and also with different legitimate emails. In this way, the *TREC 2007 Public Corpus* [8], which contains 75,419 messages, presents as a good choice in order to extend this study and our future work.

5. REFERENCES

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proceedings of the workshop on Machine Learning in the New Information Age*, pages 9–17, 2000.

[2] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[3] C. Bishop. *Pattern recognition and machine learning*. Springer New York., 2006.

[4] A. Bratko, B. Filipič, G. Cormack, T. Lynam, and B. Zupan. Spam filtering using statistical data compression models. *The Journal of Machine Learning Research*, 7:2673–2698, 2006.

[5] B. Burton. Spamprobe-bayesian spam filtering tweaks. In *Proceedings of the Spam Conference*, 2003.

[6] P. Chirita, J. Diederich, and W. Nejdl. MailRank: using ranking for spam detection. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 373–380. ACM, 2005.

[7] Y. Chiu, C. Chen, B. Jeng, and H. Lin. An Alliance-Based Anti-spam Approach. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 4, pages 203–207. IEEE, 2007.

[8] G. Cormack. TREC 2007 spam track overview. In *Sixteenth Text REtrieval Conference (TREC-2007)*, 2007.

[9] L. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome research*, 9(11):1106–1115, 1999.

[10] T. Jagatic, N. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[11] J. Kent. Information gain and a general measure of correlation. *Biometrika*, 70(1):163, 1983.

[12] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145, 1995.

[13] V. Kumar. An introduction to cluster analysis for data mining. *Computer Science Department, University of Minnesota, USA*, 2000.

[14] J. Mason. Filtering spam with spamassassin. In *HEANet Annual Conference*, 2002.

[15] E. Raymond. Bogofilter: A fast open source bayesian spam filters, 2005.

[16] G. Robinson. A statistical approach to the spam problem. *Linux J.*, 2003:3, March 2003.

[17] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos, and P. Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6(1):49–73, 2003.

[18] G. Salton and M. McGill. *Introduction to modern information retrieval*. McGraw-Hill New York, 1983.

[19] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[20] I. Santos, C. Laorden, X. Ugarte-Pedrero, B. Sanz, and P. G. Bringas. Anomaly-based spam filtering. In *Proceedings of the 6^{th} International Conference on Security and Cryptography (SECRYPT)*, pages 5–14, 2011.

[21] G. Schryen. A formal approach towards assessing the effectiveness of anti-spam procedures. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 6, pages 129–138. IEEE, 2006.

[22] W. Wilbur and K. Sirotkin. The automatic identification of stop words. *Journal of information science*, 18(1):45–55, 1992.

[23] L. Zhang, J. Zhu, and T. Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004.