An anti-spam prototype

Otávio A. S. Carpinteiro, Diego P. Almeida, Edmilson M. Moreira
Research Group on Systems and Computer Engineering
Federal University of Itajubá
Av. BPS 1303, Itajubá, MG, 37500–903, Brazil
E-mails: {otavio,edmarmo}@unifei.edu.br, digunifei@gmail.com

Technical Report

IESTI 001

May 2015

Contents

List of Figures

List of Tables

Abstract

Anti-spam systems are software systems which filter spam e-mails. Spam e-mails are nowadays a serious problem which causes high losses to the institutions. This report proposes a new anti-spam prototype made up by three stages. The first, the pre-processing, analyses the e-mails to search for known spam patterns, as well as performs eliminations or replacements to simplify the e-mails and to make them uniform. The second stage, the feature selection, identifies the most relevant features of the e-mails. The third stage, the classification, consists in an artificial neural model — the multilayer perceptron — to classify the e-mails. The anti-spam prototype is exhaustively tested on three public corpora — SpamAssassin, LingSpam and Trec 2007 — available in the Internet. The prototype performance is assessed according to the percentage of correct classifications in both e-mail classes — legitimate (ham) and spam. It is also assessed the time spent in training and testing of the neural model. The results obtained are very promising. The anti-spam prototype has very good performance on the three corpora.

Keywords: anti-spam systems · computer networks · machine learning

1 Introduction

Anti-spam (AS) systems are software systems which filter spam e-mails. Spam e-mails are unsolicited electronic messages posted blindly to many recipients, usually for commercial advertisement. Spam is a problem which causes large losses to institutions, for the number of spam e-mails circulating on computer networks is high.

The research on AS systems has received significant contributions not only by the scientific community but also by companies which develop such systems. As a result, AS systems have become gradually more complex. Despite the increasing complexity, AS systems still fail to filter both known and novel forms of spam e-mails.

The research on AS systems is currently carried out in several research centers around the world, as is evidenced by some of the recent surveys in the area [7, 15, 26]. The research on AS systems is also relevant, for it supplies the companies with mechanisms to avoid the misuse of their computational and networking resources, avoiding, therefore, considerable financial losses.

The works in the literature suggest the application of learning models to AS systems. Such models, which include Bayesian [22, 35], neural [22, 9], and kernel-based [1] models, are responsible for the classification of the e-mails.

Whatever the model implemented, however, current anti-spam systems still share the same failing, which is, the number of false positives and false negatives generated is high[1]. These numbers suggest, therefore, that AS systems should not only implement learning models to classify e-mails, but also implement some methods, such as the treatment of the information inside the e-mails, in order to improve the performance of these models.

This report aims at proposing and assessing a new AS prototype made up by three stages. The three-stage AS prototype is presented in Figure 1.

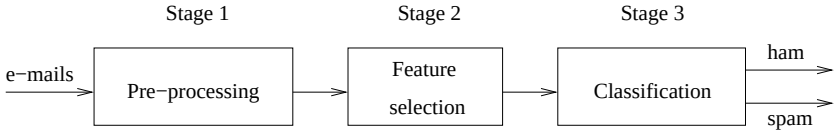

Figure 1: The three-stage AS prototype

[1]The classification of a legitimate mail as spam, and of spam as a legitimate mail is referred to as *false positive*, and *false negative*, respectively.

The first stage pre-processes the e-mails to search for known spam patterns. It also performs eliminations or replacements to simplify the e-mails and to make them uniform. The second, through feature selection methods, identifies the main features of the e-mails. The third stage consists in a neural model — the multilayer perceptron (MLP) — to classify the e-mails. The division of the AS prototype into three stages reduces its complexity and increases the performance of the neural model on the classification of e-mails.

The next sections present the pre-processing and feature selection methods proposed, the architecture of the MLP, the experiments performed, the results, the related work as well as the conclusion and some directions for future work.

2 E-mail pre-processing

Three public e-mail corpora were employed in the experiments — SpamAssassin [29], Ling-Spam [20], and Trec 2007 [10]. These corpora include both ham[2] and spam e-mails. The e-mails of the SpamAssassin corpus are complete and kept in their original form. The e-mails of the Ling-Spam corpus, in its turn, are incomplete. Their headers were removed for privacy sake. The main characteristics of the Trec corpus is the presence of image spam e-mails.

The e-mails from the three corpora were pre-processed to search for patterns commonly employed in spam e-mails, and to perform eliminations or replacements in order to simplify the e-mails, making them simpler and more uniform. Many pre-processing operations were realized on the text, images, and on HTML tags.

2.1 Pre-processing on text and images

Twelve pre-processing operations were performed on the texts and images of the e-mails. They were defined on the basis of a careful analysis performed on the contents and on the attachments of the ham and spam e-mails. The operations are described below.

1. Letters were converted into lower case. Thus, for instance, the various forms "*Home*", "*hOme*", "*hoMe*" and "*hOMe*" are represented by a unique word "*home*".
2. Accent marks were removed from the accented letters.
3. One- or two-letter words were ignored.

[2]Legitimate e-mails are also refered to as *ham e-mails*.

4. Only subject and body were considered. The rest of the header was not taken into account for it is easily manipulated by spammers.

5. Contents of e-mails may be presented in both HTML and text forms. Whenever both forms occur, the text form was discarded.

6. Images were removed. A specific tag *!_IN_IMG* was included to replace each image.

7. Attachments were removed for two reasons. First, it would be time consuming to process attachments with different file formats. Second, it was observed they hardly add new evidence for the classification of the e-mails. A specific tag *!_ATTACH_TYPE?name_attachment* was included to replace each attachment.

8. Words with twenty or more characters were discarded. A specific tag *!_BIGTEXT* was included to replace each one of them.

9. URL addresses, e-mail addresses, currency, and percentage were converted respectively into the specific tags *!_LINK*, *!_EMAIL*, *!_MONEY*, and *!_PERCENTAGE*.

10. Numbers in e-mail subjects are, to some degree, suspicious. A specific tag *!_NUMBER_SUBJECT* was included to replace each one of them.

11. Words with four or more letters separated by spaces were removed. Obviously, this pre-processing operation took precedence over the third operation, and was performed before the latter. A specific tag *!_HIDEWORDS* was included to replace each one of them. For instance, the word "*v i a g r a*" was replaced by the tag *!_HIDEWORDS*.

12. Words with delimiting characters or numbers are, to a large extent, suspicious. So, each one of them was replaced by the specific tag *!_HIDEWORDS* to signal this suspicion. For instance, the words *vi@gra* and *v1agra* were replaced by tags *!_HIDEWORDS*.

2.2 Pre-processing on HTML tags

HTML tags were divided into three categories, according to the relevance of the information they enclose. They were processed according to the category which they belong to. Table 1 presents some of the seventy-nine tags processed, as well as their respective categories.

Tags in the first category are related, in the vast majority, to the description of the document. So, they were totally discarded, that is, the tags, their attributes, and the contents they enclose were completely removed. The tag *title*, for instance, is employed to display the contents it encloses in the navigation

Table 1: Some HTML tags, and their categories

Tag	Category	Tag	Category
a	3	html	2
abbr	2	i	2
acronym	2	img	3
b	2	input	3
base	3	ins	2
blockquote	3	kbd	2
body	2	label	2
br	2	li	2
button	3	map	3
caption	2	marquee	1
col	2	ol	2
comment	1	option	2
del	2	p	2
em	2	select	2
font	3	style	1
form	3	table	2
frame	2	textarea	2
h1–h6	2	title	1
head	2	tr	2
hr	2	var	2

bar of the browsers. Thus, the block "*<title> contents </title>*" was totally discarded during pre-processing.

Tags in the second category present contents parcially significant to the classification of e-mails. Therefore, they only had their attributes removed during pre-processing. Moreover, each one of these tags was replaced by another specific tag. For instance, the block "* contents *" was converted into "*!_IN_P contents*" during pre-processing.

Tags in the third category, in its turn, present contents totally significant to the classification of e-mails. So, they were processed in their entirety except for the parameters of their attributes which were removed. More, each one of these tags was replaced by another specific tag. For instance, the block "*<form action="results.php"> contents </form>*" was converted into "*!_IN_FORM action contents*" during pre-processing.

Such as in the preprocessing operations on texts and images, the preprocessing operations on HTML tags were defined on the basis of a careful analysis performed on the e-mails.

3 Feature selection methods

Feature selection consists in extracting the most relevant features from a information set. Considering the information set as being the whole set of e-mails, the features consist then of the e-mail words, images, HTML tags and attributes which were pre-processed.

Feature selection methods are widely employed in text categorization [33]. In particular, those methods may also be employed to weight the relevance of features of e-mails for the two classes — ham and spam.

Feature selection aims at reducing the complexity of the task. Without it, the e-mails would be represented by patterns of higher dimensionality. As a consequence, the neural model would spend greater computational time to classify the e-mails.

Three feature selection methods were employed in the experiments — frequency distribution (FD), chi-square (Chi) distribution, and mutual information (MI). They were chosen for their popularity and for having presented good results in the literature. The three methods are described below.

3.1 Frequency distribution

Frequency distribution (FD) measures the degree of occurrence of an element w in a set C. If w is a feature, the frequency distribution of the feature w is given by

$$FD(w) = \frac{N[w \in \{ham, spam\}]}{T} \tag{1}$$

where $N[w \in \{ham, spam\}]$ is the number of occurrences of feature w in the classes $\{ham, spam\}$, and T the total number of features in those classes.

It is worth noticing that the FD method does not weight the relevance of a feature for a class, but only the amount of its occurrence. By this reason, in some cases, a feature with low occurence in the e-mail corpus, even if it is relevant for the classification of a certain e-mail, may not be taken into account.

The features with the highest values of FD are then selected. Each selected feature is represented by one input unit of the neural model, normalized in the interval "*[0, 1]*" by the maximum norm.

3.2 Chi-square distribution

Chi-square (Chi) distribution measures the degree of dependence between an element e and a set S [24]. If w is a feature, and C a set of two classes — ham and spam —, the chi-square distribution of the feature w is given by

$$Chi(w) = P(ham) \cdot Chi(w, ham) + P(spam) \cdot Chi(w, spam) \tag{2}$$

where $P(ham)$ and $P(spam)$ are the probabilities of occurrence of ham and spam e-mails, respectively. The chi-square distribution for the feature w and class c is given by

$$Chi(w, c) = \frac{N(kn - ml)^2}{(k + m)(l + n)(k + l)(m + n)} \tag{3}$$

where k is the number of e-mails, within class c, which contain the feature w; l is the number of e-mails, within class $\overline{c}$, which contain the feature w; m is the number of e-mails, within class c, which do not contain the feature w; n is the number of e-mails, within class $\overline{c}$, which do not contain the feature w; and N is the total number of e-mails within class c.

The features with the highest values of chi-square distribution are then selected. Each selected feature is represented by one input unit of the neural model, normalized in the interval "*[0, 1]*" by the maximum norm.

3.3 Mutual information

Mutual information (MI) is a basic method in information theory [12]. If w is a feature, the mutual information of the feature w is given by

$$MI(w) = \sum_{f=\{w,\overline{w}\}} \sum_{c=\{ham,spam\}} P(f, c) \log_2 \frac{P(f, c)}{P(f) \cdot P(c)} \tag{4}$$

where $P(f, c) = P(f \cap c) = P(c) \cdot P(f \mid c)$ is the probability that both f and c occur. The features with the highest values of MI are then selected. Each selected feature is represented by one input unit of the neural model, normalized in the interval "*[0, 1]*" by the maximum norm.

4 Multilayer perceptron

Multilayer perceptrons (MLPs) have been widely employed in pattern recognition problems [4]. In this class of problems, they present several advantages over other models. For instance, they are simple, fast in classification tasks, make use of parallel processing and present graceful degradation. Moreover, by inductive learning, they establish themselves the function which maps the set of inputs on the set of outputs [16].

The main advantage, however, is the fact that MLPs are capable of generalization. They can produce correct outputs even on inputs that were never presented to them during training. The generalization property is particularly interesting in the domain of anti-spam systems, for novel forms of spam e-mails emerge continuosly.

The multilayer perceptron (MLP) used in the experiments holds from five to one hundred input units, according to the dimension of the input vector, and two output units. It was trained to display activation values *(0 1)* and *(1 0)* in the output units when the units in the input layer are representing, respectively, a ham and a spam e-mail.

The MLP holds 11 and 17 units in the first and second hidden layers, respectively. The hidden units hold the tangent sigmoid (tansig) activation function, whilst the output units hold the sigmoid function.

Training takes place on an epoch-by-epoch basis through the backpropagation learning algorithm [16]. It is performed through cross validation. Therefore, it is halted whenever the total error increases on the validation set. The initial weights are randomly assigned.

5 Methodology used in the experiments

The experiments were carried out on three different corpora — SpamAssassin, Ling-Spam and Trec 2007 —, all public, with real e-mails and available on the Internet.

The SpamAssassin corpus [29] is divided into five parts — two directories with spam e-mails and three with ham e-mails — totaling 6037 e-mails, of which 1844 are spams and 4193 hams. The original e-mail headers were preserved, except for some alterations in the address only, made for privacy reasons. As it presents e-mails retrieved from varied sources, in which the original content was preserved (including HTML contents), the SpamAssassin corpus constitutes a challenging corpus, which strives to satisfy the heterogeneity found by anti-spam systems in real-life situations. For this reason, it was chosen as the e-mail

corpus in most of the experiments.

The Ling-Spam corpus [20] is composed of 2893 e-mails, of which 481 are spams and 2412 hams. Androutsopoulos et al. [2] describe the Ling-Spam corpus as being a composition of spam messages, received from a personal e-mail mailbox, and of legitimate messages, sent to a discussion list. Two important deficiencies of this corpus are the fact that the spam e-mails originate from a single user profile and the fact that the attachments and HTML tags were removed from the e-mails, making the classification task simpler.

The Trec 2007 corpus [10] has 75288 e-mails, of which 25214 are hams and 50074 spams, which were collected between 4/8/2007 and 7/6/2007 from a private e-mail server. The main characteristic of the Trec corpus is the presence of image spam e-mails.

In all the experiments, input patterns (vectors) with no more than 100 features selected by the feature selection methods were used. Hence an e-mail may not contain any of the selected features, and consequently, the input pattern that represents it will be the null pattern.

In the MLP neural model training, these null patterns must not be present. Therefore, the training and validation sets never included null patterns. In the test set, however, the null patterns were reinserted.

The training, validation and test sets are balanced, with each of them having an equal number of ham and spam e-mails. Therefore, the class — spam or ham — with the smallest number of patterns has part of its patterns duplicated. The patterns to be duplicated are chosen randomly.

The input patterns are normalized in the interval *[0, 1]* by the maximum norm. In the experiments, the quantity of features present in the input patterns varied, every five features, until it reached the maximum of 100 features. The quantity of patterns in the training, validation and test sets follows the proportion of 40%, 20% and 40% respectively.

For statistical purposes, each experiment was carried out ten times, and in each one of them, the neural model was trained with different initial weights, generated randomly. Therefore, for each experiment, the performance of the neural model is measured through the simple mean of the 10 errors obtained in the classification of ham and spam e-mails, as well as through the dispersion of these errors in relation to the mean.

The number of patterns used in each of the three sets — training, validation and test —, as well as the number of null and duplicated patterns, is reported in each experiment described below. The results obtained with or without the use of each one of the three feature selection methods, as well as the computational time spent on the training and testing of the neural model, are reported as well.

A computer with the hardware and software configurations displayed in the Table 2 was used in all the experiments.

Table 2: Configuration of the computer used in the experiments

PC Model	Sony Vaio® VGN-FS960
Processor	Intel Pentium® M 1.73GHz
Memory	1.5 GB
Operating System	Windows 7 Ultimate® 32-bit
JVM	Java 6 Update 17

6 Experiments on the SpamAssassin corpus

The three first experiments described below were carried out to assess the importance of the first two stages — pre-processing and feature selection — in the performance of the anti-spam prototype. Experiment 4 evaluated the joint use of pre-processing techniques and feature selection methods.

6.1 Experiment 1

Experiment 1 did not make use of the first two stages — pre-processing and feature selection — of the anti-spam prototype. Table 3 details the training, validation and test sets used in the experiment.

Table 3: Ham and spam patterns in the training, validation and test sets of experiment 1

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	81	40	0	0	0	41	162
Validation	41	20	0	0	0	21	82
Test	81	39	4135	1600	0	42	5897

As may be seen in Table 3, as no method is used to select the most significant features, a large quantity of null patterns is obtained, reducing the training and validation sets to only 162 and 82 patterns, respectively. In the test set, as mentioned previously, the null patterns were reinserted.

Tables 4 and 5 present, respectively, the results obtained in the experiment and the computational times spent on the training and testing of the neural

model. Figure 2 presents, in graphic form, the performance of the neural model in the classification of the ham and spam patterns.

Table 4: Results of experiment 1

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	28.56 ± 0.00	*100.00* ± 0.00
10	28.58 ± 0.00	100.00 ± 0.00
15	28.58 ± 0.00	100.00 ± 0.00
20	35.76 ± 7.13	97.03 ± 2.83
25	28.63 ± 0.00	99.85 ± 0.00
30	28.82 ± 0.00	99.82 ± 0.00
35	28.82 ± 0.00	99.82 ± 0.00
40	35.93 ± 7.12	96.99 ± 2.83
45	*92.88* ± 7.12	74.36 ± 2.83
50	28.85 ± 0.00	99.83 ± 0.01
55	28.85 ± 0.00	99.82 ± 0.00
60	57.31 ± 28.46	88.50 ± 11.31
65	28.89 ± 0.00	99.74 ± 0.00
70	28.90 ± 0.01	99.83 ± 0.06
75	92.83 ± 7.13	74.70 ± 2.89
80	64.49 ± 35.48	85.81 ± 14.00
85	64.69 ± 35.18	85.62 ± 13.77
90	43.51 ± 14.66	93.90 ± 5.99
95	64.78 ± 35.14	85.69 ± 13.70
100	71.82 ± 28.14	82.91 ± 10.99

Table 5: Computational times (in seconds) spent on the training and testing in experiment 1

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	0.83	*0.07*
10	0.92	0.07
15	0.77	0.08
20	0.74	0.09
25	0.99	0.08
30	0.91	0.08
35	1.27	0.09
40	0.94	0.13
45	1.03	0.10
50	1.03	0.09
55	0.90	0.10
60	0.91	0.10
65	1.03	0.10
70	0.76	0.10
75	0.75	0.10
80	0.79	0.10
85	0.76	0.10
90	*0.71*	0.10
95	0.87	0.14
100	0.73	0.11

Even though the spam classification results indicate highly satisfactory values, especially with the use of input patterns with a lower quantity of features, a high incidence of false positives (hams classified as spams) may be observed.

The low quality of the classification presented is due mainly to the fact that the features chosen to form the inputs patterns are not significant. The error behaves in an absolutely random manner, not presenting reduction with the increase in the number of features present in the patterns, as shown in Figure 2. Another negative point may be observed in relation to the dispersion of the measures, presenting accentuated variations.

Lastly, it is important to emphasize that the comparison of the computational time of training of the neural model in relation to that of the other experiments, which are presented below, is not reasonable, owing to the low quantity of patterns existing in the training and validation sets.

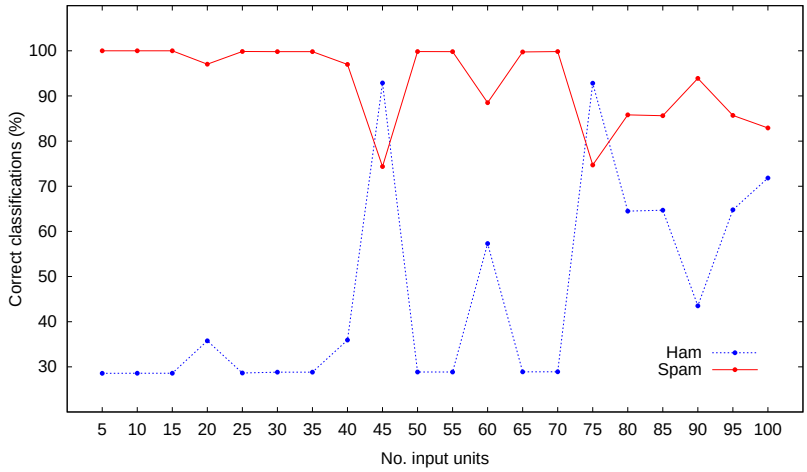

Figure 2: Performance of the neural model in the classification of the ham and spam patterns

6.2 Experiment 2

The first stage — pre-processing — of the anti-spam prototype was used in experiment 2. However, the second stage — feature selection — was not employed. Table 6 details the training, validation and test sets used in the experiment.

Table 6: Ham and spam patterns in the training, validation and test sets of experiment 2

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	390	170	0	0	0	220	780
Validation	195	85	0	0	0	110	390
Test	389	169	3219	1420	0	220	5417

Although pre-processing reduce the number of features present in the e-mails, the non-use of a feature selection method means that the choice of features is not significant, generating a considerable number of null patterns. Hence the training and validation sets were reduced to 780 and 390 patterns, respectively.

Tables 7 and 8 present, respectively, the results obtained in the experiment and the computational times spent on the training and testing of the neural model.

Table 7: Results of experiment 2

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	66.69 ± 33.29	83.30 ± 16.70
10	80.02 ± 19.97	76.62 ± 10.02
15	60.05 ± 26.62	86.59 ± 13.32
20	33.43 ± 0.00	*99.91* ± 0.00
25	*99.63* ± 0.00	67.69 ± 0.02
30	99.42 ± 0.01	68.08 ± 0.04
35	92.81 ± 6.62	71.27 ± 3.32
40	99.38 ± 0.03	68.17 ± 0.16
45	99.37 ± 0.12	68.23 ± 0.32
50	99.26 ± 0.05	68.51 ± 0.07
55	99.26 ± 0.07	68.49 ± 0.18
60	99.24 ± 0.05	68.68 ± 0.03
65	98.76 ± 0.06	69.16 ± 0.14
70	98.73 ± 0.05	69.10 ± 0.08
75	98.73 ± 0.10	69.61 ± 0.21
80	92.42 ± 6.07	72.45 ± 2.69
85	74.53 ± 24.18	80.39 ± 10.87
90	92.42 ± 6.22	72.53 ± 2.80
95	74.51 ± 23.96	80.44 ± 10.64
100	50.77 ± 11.97	91.03 ± 5.39

Table 8: Computational times (in seconds) spent on the training and testing in experiment 2

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	1.14	*0.06*
10	1.15	0.06
15	1.21	0.07
20	1.43	0.07
25	1.08	0.07
30	1.13	0.07
35	1.53	0.08
40	1.35	0.08
45	1.20	0.08
50	*1.06*	0.08
55	1.25	0.09
60	1.39	0.09
65	2.03	0.09
70	1.74	0.09
75	2.01	0.09
80	1.25	0.10
85	1.38	0.10
90	1.43	0.10
95	1.44	0.11
100	1.43	0.11

The results obtained in experiment 2 are better than those of experiment 1, although, as shown in Figure 3, the error behaviour remain random, not presenting a reduction with the increase in the number of features present in the patterns. Moreover, the dispersion of measures also remains significant.

Once again, just like in experiment 1, the comparison of the computational time of training of the neural model in relation to that of the other experiments is not reasonable, owing to the low quantity of patterns existing in the training and validation sets.

6.3 Experiment 3

The first stage — pre-processing — of the anti-spam prototype was not used in experiment 3. However, the three feature selection methods of the second stage were employed. The results are presented in three subsections, one for each one of the methods — FD, Chi-square and MI — used.

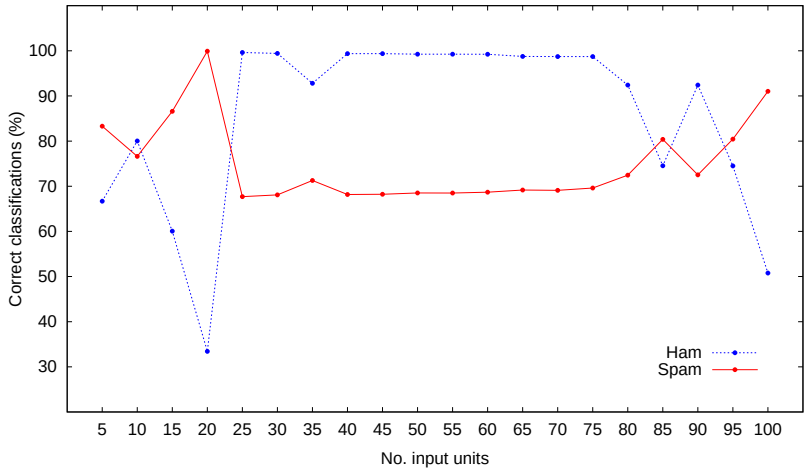

Figure 3: Performance of the neural model in the classification of the ham and spam patterns

6.3.1 Experiment 3 — Method FD

Table 9 details the training, validation and test sets used in the experiment.

Table 9: Ham and spam patterns in the training, validation and test sets of experiment 3 — FD method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	1659	724	0	0	0	935	3318
Validation	829	362	0	0	0	467	1658
Test	1659	723	46	35	0	936	3399

Observing just the profile of the sets, one may notice the difference in relation to those of the previous experiments. This time, the number of null patterns was significantly reduced, reinforcing the hypothesis that the use of feature selection methods is important for the efficient choice of e-mail features.

Tables 10 and 11 present, respectively, the results obtained with the use of the FD method in the experiment and the computational times spent on the training and testing of the neural model.

The better selection of the e-mail features was reflected positively in the results obtained. The classification error decreases with the increase in the

Table 10: Results of experiment 3 — FD method

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	79.02 ± 1.32	85.64 ± 1.74
10	84.98 ± 1.22	89.43 ± 1.18
15	84.65 ± 1.21	89.81 ± 1.22
20	88.84 ± 1.70	90.51 ± 1.25
25	88.55 ± 1.38	91.58 ± 1.00
30	90.45 ± 1.72	90.66 ± 1.57
35	93.12 ± 1.20	91.35 ± 0.85
40	93.13 ± 2.22	91.14 ± 1.61
45	92.78 ± 1.10	92.39 ± 0.57
50	93.03 ± 1.09	91.64 ± 1.12
55	92.41 ± 1.70	92.06 ± 1.94
60	92.02 ± 2.71	93.19 ± 1.81
65	91.26 ± 3.68	93.43 ± 1.24
70	92.84 ± 1.96	93.46 ± 1.28
75	92.71 ± 2.23	93.45 ± 0.98
80	93.34 ± 2.19	93.48 ± 1.51
85	93.18 ± 2.49	93.47 ± 1.73
90	93.86 ± 1.91	93.50 ± 0.82
95	*94.10* ± 1.52	93.56 ± 1.12
100	93.40 ± 2.31	*93.90* ± 0.81

Table 11: Computational times (in seconds) spent on the training and testing in experiment 3 — FD method

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	6.04	*0.05*
10	4.14	0.05
15	3.55	0.05
20	3.46	0.06
25	4.40	0.06
30	3.74	0.06
35	4.06	0.06
40	3.93	0.06
45	3.72	0.06
50	3.79	0.07
55	*3.23*	0.06
60	3.47	0.06
65	3.65	0.07
70	3.83	0.07
75	3.89	0.07
80	3.61	0.07
85	3.69	0.08
90	3.74	0.08
95	3.95	0.08
100	3.95	0.08

number of features present in the patterns and the performances of the neural model in the classification of spam and ham e-mails are similar. In addition, the dispersion of measures decreased significantly.

An important fact may be observed in the computational times spent on the training of the neural model. With a larger number of features present in the input patterns, the times spent on training should theoretically increase, owing to the increase in the number of connections in the architecture of the neural model. However, the computational times initially decrease to later stabilize, because despite the greater computational processing load by training epoch, the convergence of the training improves, i.e., the neural model needs a smaller number of epochs for its training.

The computational time spent on testing the neural model presents the expected behaviour. The larger the number of features present in the input patterns, the longer the time the neural model takes to produce the response.

6.3.2 Experiment 3 — Method Chi-Square

Table 12 details the training, validation and test sets used in the experiment.

Table 12: Ham and spam patterns in the training, validation and test sets of experiment 3 — Chi-Square Method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	1467	716	0	0	0	751	2934
Validation	733	358	0	0	0	375	1466
Test	1467	716	526	54	0	751	3514

There is an increase in the number of null patterns in relation to the number produced by the FD method (section 6.3.1). This suggests that the choice of the most significant features, made by the FD method to compose the input patterns, was more accurate and comprehensive in the SpamAssassin corpus. However, this condition alone does not imply greater efficiency of the FD method, as the results produced by the application of this method are also required to be superior to those of the Chi-Square method.

Table 13 presents the results obtained with the use of the Chi-Square method in the experiment. The table 14 shows the computational times spent on the training and testing of the neural model.

The results obtained show that although the Chi-square method present a performance inferior to that presented by the FD method in the correct classification of spam e-mails, it presents a better performance in relation to false positives, i.e., presents a smaller error in the classification of ham e-mails.

The computational times of training of the neural model are shorter than those obtained with the FD method (section 6.3.1), because the quantity of null patterns generated by the Chi-Square method is greater.

6.3.3 Experiment 3 — Method MI

Table 15 details the training, validation and test sets used in the experiment. Note that the number of null patterns produced by the MI method is lower than the number produced by the Chi-Square method. It is, however, higher than the number of null patterns produced by the FD method.

Tables 16 and 17 present, respectively, the results obtained with the use of the MI method in the experiment and the computational times spent on the training and testing of the neural model.

Table 13: Results of experiment 3 — Chi-Square Method

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	94.14 ± 1.17	86.04 ± 1.15
10	*97.02* ± 0.67	86.96 ± 0.71
15	94.90 ± 1.06	89.93 ± 1.05
20	94.90 ± 1.20	90.35 ± 1.54
25	95.61 ± 1.45	90.25 ± 0.90
30	95.33 ± 0.35	90.11 ± 0.87
35	95.00 ± 0.87	90.40 ± 0.82
40	94.98 ± 0.53	92.60 ± 1.57
45	93.36 ± 3.11	92.64 ± 1.29
50	95.53 ± 0.82	91.82 ± 0.86
55	95.39 ± 0.74	92.53 ± 1.15
60	95.53 ± 0.72	92.37 ± 0.77
65	95.62 ± 0.57	92.52 ± 0.99
70	95.69 ± 0.53	92.08 ± 0.95
75	95.52 ± 0.75	92.38 ± 0.69
80	95.77 ± 0.68	92.60 ± 1.05
85	95.78 ± 0.55	92.50 ± 0.92
90	95.02 ± 1.00	92.82 ± 0.73
95	95.42 ± 0.37	*92.85* ± 0.81
100	95.61 ± 0.50	92.40 ± 1.09

Table 14: Computational times (in seconds) spent on the training and testing in experiment 3 — Chi-Square method

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	3.62	*0.05*
10	3.36	0.06
15	3.27	0.05
20	3.03	0.06
25	3.93	0.06
30	3.33	0.06
35	3.42	0.06
40	2.92	0.06
45	3.15	0.06
50	*2.74*	0.07
55	2.98	0.07
60	3.13	0.07
65	2.84	0.07
70	3.01	0.07
75	3.09	0.07
80	3.07	0.08
85	3.28	0.07
90	3.31	0.08
95	3.26	0.08
100	3.33	0.08

Table 15: Ham and spam patterns in the training, validation and test sets of experiment 3 — MI method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	1609	719	0	0	0	890	3218
Validation	804	359	0	0	0	445	1658
Test	1609	719	171	47	0	890	3436

Analyzing the results, one may notice that the MI method, among the three feature selection methods, is the one that presents the best performance, both in the correct classification of ham and of spam e-mails.

The times spent on the training of the neural classifier appear similar, regardless of the number of features in the input pattern. As mentioned previously,

Table 16: Results of experiment 3 — MI method

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	90.40 ± 0.18	82.31 ± 0.26
10	91.42 ± 0.39	88.29 ± 0.22
15	91.44 ± 0.58	88.23 ± 0.72
20	94.14 ± 0.63	88.98 ± 0.51
25	93.86 ± 0.73	88.96 ± 0.54
30	93.82 ± 1.38	90.80 ± 0.87
35	93.38 ± 1.12	91.16 ± 0.88
40	94.18 ± 0.49	91.59 ± 0.73
45	91.07 ± 2.57	93.98 ± 1.28
50	91.72 ± 3.10	94.32 ± 1.55
55	92.86 ± 2.54	93.66 ± 1.55
60	92.80 ± 3.04	94.02 ± 1.35
65	93.27 ± 1.93	94.01 ± 1.49
70	95.07 ± 0.60	93.40 ± 0.97
75	93.15 ± 2.19	*94.77* ± 0.88
80	95.40 ± 0.29	94.19 ± 0.68
85	94.60 ± 2.14	94.07 ± 1.06
90	95.59 ± 0.48	93.82 ± 0.54
95	*95.72* ± 0.62	93.93 ± 0.40
100	95.71 ± 0.42	93.92 ± 0.26

Table 17: Computational times (in seconds) spent on the training and testing in experiment 3 — MI method

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	3.16	*0.06*
10	3.56	0.06
15	3.44	0.06
20	4.60	0.06
25	4.36	0.06
30	4.42	0.06
35	4.32	0.07
40	3.25	0.06
45	3.14	0.06
50	3.15	0.06
55	3.15	0.07
60	*3.11*	0.06
65	3.35	0.07
70	4.11	0.07
75	3.27	0.07
80	3.20	0.07
85	3.74	0.07
90	4.03	0.08
95	3.57	0.08
100	3.67	0.08

larger numbers of features are offset by better convergences in the training. In relation to the computational times spent on the testing of the neural model, the expected behaviour occurs. The larger the number of features in the input pattern, the longer the neural model takes to produce the response.

6.4 Experiment 4

Through the results of experiments 1 (section 6.1), 2 (section 6.2), and 3 (section 6.3), it is verified that the use of the feature selection methods is important for the correct classification of spam and ham e-mails. Moreover, despite the good results obtained in experiment 3, experiment 2 suggests that these may be further improved through the use of the pre-processing techniques. Therefore, the experiment 4 made use of both the pre-processing techniques and the feature selection methods.

6.4.1 Experiment 4 — Method FD

Table 18 details the training, validation and test sets used in the experiment.

Table 18: Ham and spam patterns in the training, validation and test sets of experiment 4 — FD method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	1677	736	0	0	0	941	3354
Validation	839	368	0	0	0	471	1678
Test	1677	737	0	3	0	940	3357

It may be seen in the Table 18 that the use both of pre-processing and of the FD method produces only three null patterns, a low number when compared with the numbers produced by the previous experiments.

Tables 19 and 20 present, respectively, the results obtained with the use of the FD method in the experiment and the computational times spent on the training and testing of the neural model.

The more accurate results in relation to those of experiment 3, with the FD method, suggest that the use both of pre-processing and of the feature selection methods significantly reduces the number of incorrect classifications of spam and ham e-mails. It is important to emphasize that, besides the mean error, the dispersions of measures also decreased.

The computational times spent on the training of the neural model remain practically constant. Just like in experiment 3 with the MI method (section 6.3.3), larger numbers of features in the input pattern are offset by better convergences in the training. The computational times spent on the testing of the neural model behave as expected. The larger the number of features in the input pattern, the longer the neural model takes to produce the response.

6.4.2 Experiment 4 — Method Chi-Square

Table 21 details the training, validation and test sets used in the experiment.

Just like in experiment 3 with the Chi-Square method (section 6.3.2), one may notice an increase in the number of null patterns in relation to the number produced by the FD method (section 6.4.1). This suggests that the choice of the most significant features, made by the FD method to compose the input patterns, was more accurate and comprehensive in the SpamAssassin corpus.

Tables 22 and 23 present, respectively, the results obtained with the use of

Table 19: Results of experiment 4 — FD method

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	92.04 ± 1.70	88.19 ± 1.32
10	93.43 ± 1.54	89.13 ± 1.43
15	94.40 ± 0.77	89.21 ± 1.51
20	95.51 ± 0.89	91.59 ± 1.51
25	94.49 ± 0.68	92.55 ± 1.28
30	94.37 ± 1.96	93.52 ± 0.97
35	95.57 ± 1.33	94.61 ± 0.69
40	95.43 ± 1.53	94.59 ± 1.06
45	95.05 ± 1.64	95.12 ± 0.80
50	95.89 ± 1.08	95.44 ± 0.71
55	96.08 ± 1.29	95.07 ± 0.67
60	95.81 ± 1.75	95.49 ± 0.57
65	96.66 ± 0.69	94.83 ± 0.59
70	96.61 ± 0.66	94.73 ± 0.62
75	97.00 ± 0.70	94.69 ± 0.77
80	97.18 ± 0.47	95.60 ± 0.88
85	*97.42* ± 0.56	95.48 ± 1.03
90	97.13 ± 0.71	95.90 ± 1.15
95	97.08 ± 0.80	96.26 ± 0.85
100	96.84 ± 1.05	*96.66* ± 1.07

Table 20: Computational times (in seconds) spent on the training and testing in experiment 4 — FD method

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	6.16	0.05
10	*2.81*	0.06
15	3.54	*0.05*
20	4.09	0.06
25	3.91	0.06
30	4.00	0.06
35	4.27	0.06
40	4.05	0.06
45	3.48	0.06
50	4.15	0.07
55	3.93	0.06
60	3.81	0.07
65	4.03	0.06
70	3.92	0.07
75	4.23	0.08
80	4.14	0.07
85	4.99	0.07
90	4.03	0.08
95	4.28	0.08
100	4.32	0.08

Table 21: Ham and spam patterns in the training, validation and test sets of experiment 4 — Chi-Square Method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	1668	734	0	0	0	934	3336
Validation	834	367	0	0	0	467	1668
Test	1667	735	24	8	0	932	3366

the Chi-Square method in the experiment and the computational times spent on the training and testing of the neural model.

The results are similar to those of the previous experiment (section 6.4.1). Although the Chi-Square method has on average produced a higher error than that of the FD method in the classification of spam e-mails, presented a lower error on average in the classification of ham e-mails.

Table 22: Results of experiment 4 — Chi-Square method

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	96.53 ± 0.38	90.30 ± 0.61
10	96.19 ± 1.55	91.97 ± 0.91
15	96.18 ± 0.94	93.42 ± 1.30
20	96.65 ± 0.92	93.66 ± 0.52
25	96.66 ± 1.27	93.42 ± 0.46
30	96.63 ± 0.55	94.81 ± 0.46
35	96.62 ± 0.87	95.04 ± 0.62
40	97.38 ± 0.51	94.96 ± 0.73
45	97.36 ± 0.73	95.10 ± 0.39
50	97.80 ± 0.30	95.12 ± 0.23
55	97.83 ± 0.29	94.98 ± 0.53
60	*98.56* ± 0.34	94.80 ± 0.29
65	98.45 ± 0.29	94.93 ± 0.23
70	98.04 ± 0.56	*95.28* ± 0.78
75	97.85 ± 0.69	95.24 ± 0.62
80	98.03 ± 1.11	94.88 ± 1.08
85	98.14 ± 0.76	94.96 ± 0.67
90	98.34 ± 0.33	94.75 ± 0.40
95	98.20 ± 0.62	94.89 ± 0.65
100	98.29 ± 0.64	94.92 ± 0.41

Table 23: Computational times (in seconds) spent on the training and testing in experiment 4 — Chi-Square method

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	3.30	*0.05*
10	3.82	0.05
15	4.02	0.06
20	4.35	0.05
25	3.85	0.06
30	3.33	0.05
35	*2.81*	0.06
40	3.13	0.06
45	3.15	0.06
50	3.26	0.06
55	2.99	0.06
60	3.49	0.06
65	3.30	0.07
70	3.40	0.07
75	3.23	0.07
80	3.33	0.07
85	3.61	0.08
90	3.56	0.07
95	3.70	0.08
100	3.75	0.08

The computational times of training of the neural model are shorter than those obtained in the previous experiment (section 6.4.1), with the FD method, because the quantity of null patterns generated by the Chi-Square method is greater.

6.4.3 Experiment 4 — Method MI

Table 24 details the training, validation and test sets used in the experiment.

It may be seen in Table 24 that the use of both pre-processing and of the MI method produces only five null patterns, i.e., a low value.

Tables 25 and 26 present, respectively, the results obtained with the use of the MI method in the experiment and the computational times spent on the training and testing of the neural model.

Table 24: Ham and spam patterns in the training, validation and test sets of experiment 4 — MI method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	1677	736	0	0	0	941	3354
Validation	838	368	0	0	0	470	1676
Test	1677	736	1	4	0	941	3359

Table 25: Results of experiment 4 — MI method

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	93.16 ± 0.20	87.74 ± 0.59
10	96.22 ± 1.18	91.19 ± 0.80
15	96.13 ± 1.40	92.18 ± 1.29
20	97.29 ± 0.30	94.25 ± 1.00
25	96.91 ± 0.97	94.58 ± 1.17
30	96.72 ± 0.80	95.27 ± 0.68
35	96.47 ± 0.78	95.48 ± 1.30
40	96.40 ± 0.97	95.63 ± 1.07
45	96.85 ± 0.39	95.33 ± 0.91
50	96.76 ± 1.00	95.51 ± 0.98
55	96.80 ± 0.76	95.67 ± 0.31
60	97.66 ± 0.44	95.84 ± 0.71
65	98.19 ± 0.69	95.83 ± 0.98
70	98.82 ± 0.39	95.50 ± 0.98
75	98.84 ± 0.50	95.73 ± 0.72
80	*98.95* ± 0.40	95.90 ± 0.74
85	98.77 ± 0.31	96.15 ± 0.86
90	98.90 ± 0.41	95.98 ± 0.98
95	98.89 ± 0.54	*96.30* ± 0.92
100	98.87 ± 0.37	96.15 ± 0.62

Table 26: Computational times (in seconds) spent on the training and testing in experiment 4 — MI method

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	4.48	*0.05*
10	3.05	0.05
15	*2.72*	0.05
20	4.83	0.06
25	3.02	0.05
30	3.13	0.05
35	3.30	0.06
40	3.29	0.06
45	3.33	0.05
50	3.37	0.06
55	3.51	0.06
60	3.31	0.06
65	3.56	0.06
70	3.53	0.06
75	3.79	0.07
80	3.55	0.07
85	3.67	0.07
90	3.57	0.07
95	3.58	0.07
100	3.64	0.07

The results indicate that the MI method with the use of pre-processing presents a performance superior to both the FD and Chi-Square method also with the use of pre-processing.

6.5 Analysis of the results obtained on the SpamAssassin corpus (experiments 1 to 4)

The results obtained in experiments 1 and 2 are poor when compared with those of experiments 3 and 4. This suggests that feature selection methods are important for the correct classification of the e-mails performed by the neural model.

Figures 4 and 5 present the percentages of correct classifications of ham and spam e-mails obtained in experiment 3, according to the number of features selected in each of the three methods.

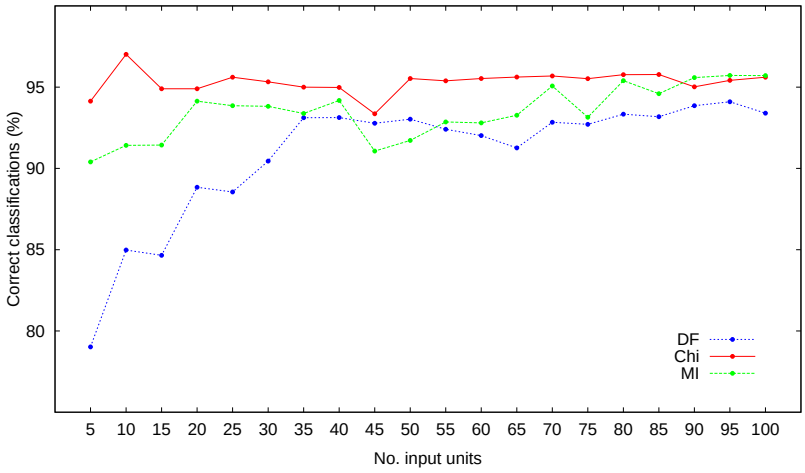

Figure 4: Comparison of the three methods in the classification of ham e-mails in experiment 3

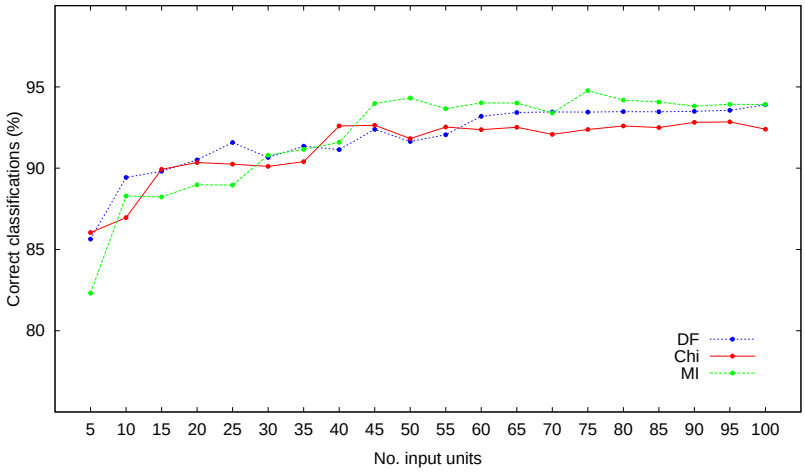

Figure 5: Comparison of the three methods in the classification of spam e-mails in experiment 3

Some conclusions may be drawn from Figures 4 and 5. Firstly, the classification of ham e-mails presents a small gain with the increase in the number of features selected and the classification of spam e-mails, in turn, presents a large gain with such an increase. Secondly, the error, both for ham e-mails and for spam, tends to stabilize from a number of around 50 features. Thirdly, out of the 3 methods, the Chi-Square method is the one which presents the most homogeneous behaviour, i.e., the least variation of error with the increase in the number of features selected. In fourth place, the MI method, in general terms, is the one which presents the best overall performance in the classification of e-mails. However, in the classification of ham e-mails, particularly, the Chi-Square method was more accurate for input patterns with up to 85 features. In fifth place, the MI method has the worst performance for classification of spam e-mails when the patterns present a reduced number of inputs. In sixth place, all the methods present satisfactory performance in terms of number of false positives.

Figures 6 and 7 present the percentages of correct classifications of ham and spam e-mails obtained in experiment 4, according to the number of features selected in each of the three methods.

The results of experiment 4 behave similarly to those of experiment 3. The only and significant difference consists in the fact that all the results of experiment 4 are superior, owing to the use of pre-processing.

Figure 8 presents the total percentages of correct classification of the e-mails in experiments 1 to 4. These percentages are calculated through the means of

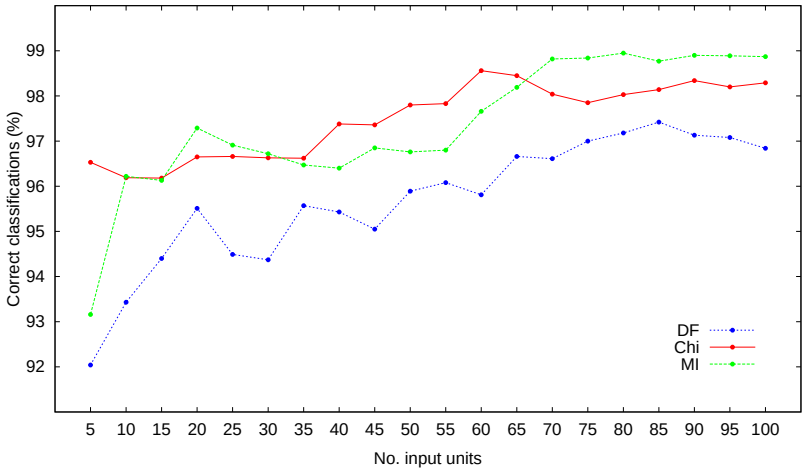

Figure 6: Comparison of the three methods in the classification of ham e-mails in experiment 4

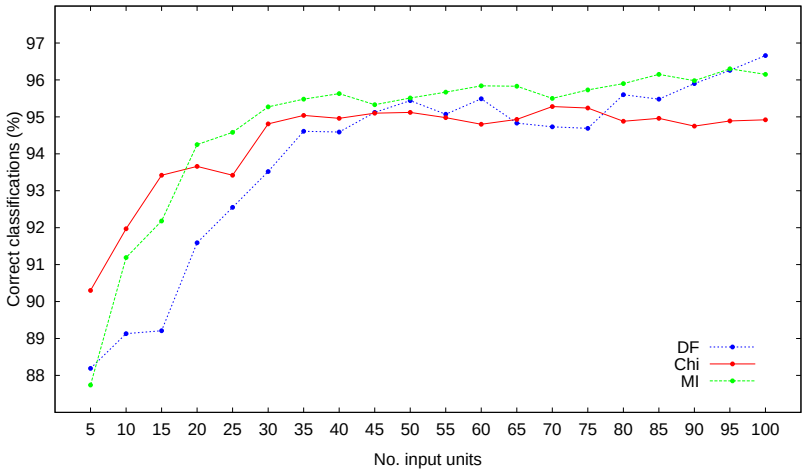

Figure 7: Comparison of the three methods in the classification of spam e-mails in experiment 4

the correct classifications obtained in the classifications of spam and ham e-mails, considering all the results with the different numbers of features present in the input patterns. For experiments 3 and 4, the feature selection method considered was MI, as it presented the best average performance of the three methods.

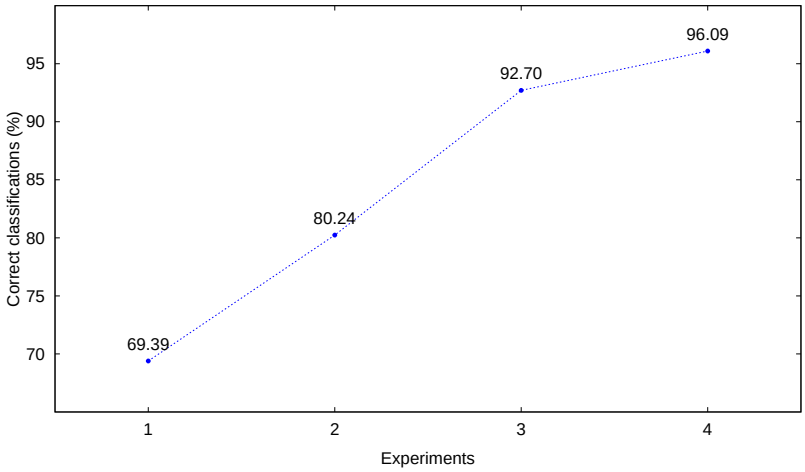

Figure 8: Total percentages of correct classification of the e-mails in experiments 1 to 4

As may be observed in Figure 8, the most significant improvement in the total percentage of correct classifications occurs in experiments 3 and 4, i.e., when the feature selection methods are employed. Moreover, through the observation of the total percentage of experiment 2 in relation to experiment 1 and of the total percentage of experiment 4 in relation to experiment 3, it is verified that the pre-processing techniques also produce a significant reduction in error, when added.

Figures 9 and 10 present, respectively, the average computational times spent by the neural model on training and testing, for input patterns containing from five to one hundred features.

By analyzing the Figure 9, it may be noticed that the average computational time spent on the training of the neural model is influenced by three factors. Firstly, by the dimension of the input pattern, i.e., by the number of features selected. Secondly, by the size of the training set. Thirdly, by the relevance of the features selected to form the input pattern, determining a faster convergence of the training or not.

The difference in the average computational times spent on training, by the neural model, between experiments 1 and 2 and between experiments 2 and 3, can be explained by the size of the training set. In fact, in experiment 1, the number of null patterns is higher than the number of null patterns in experiment 2, which, in turn, is higher than the number of null patterns in experiment 3. Hence the set of training patterns used in experiment 1 is smaller than the set used in experiment 2, which, in turn, is much smaller than that used in

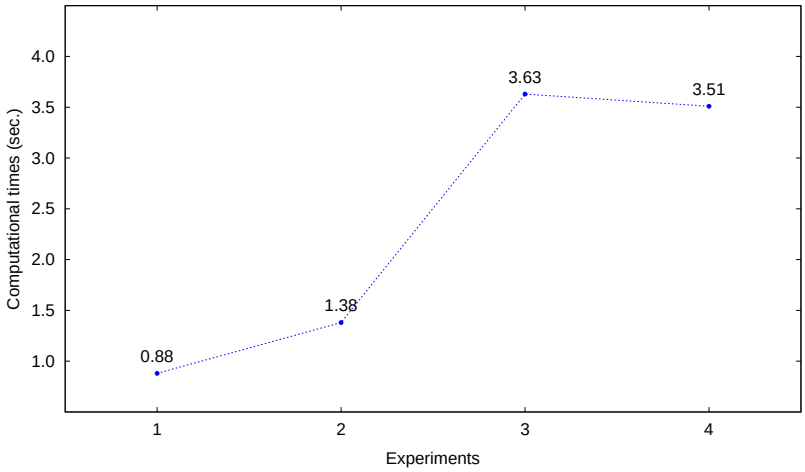

Figure 9: Average computational times spent by the neural model on the training

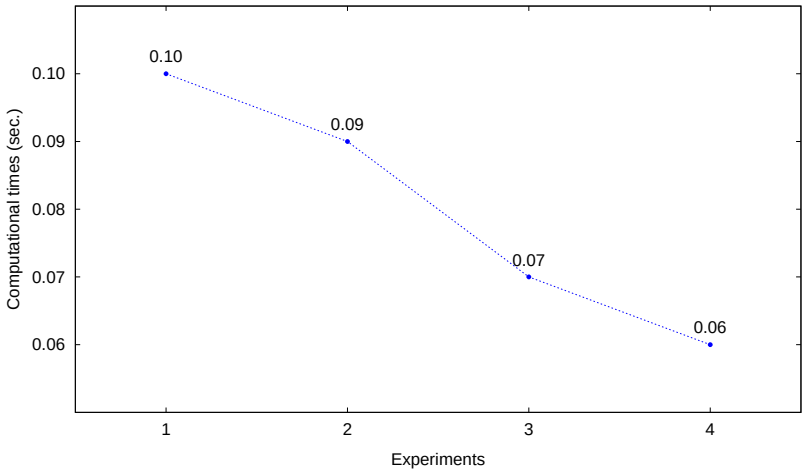

Figure 10: Average computational times spent by the neural model on the testing

experiment 3.

The difference in the average computational times spent on training, by the neural model, between experiments 3 and 4 can be explained by the relevance of the features selected to form the input pattern. Pre-processing, by eliminating

nonsignificant features and standardizing the other features, as described in section 2, reduces the total quantity of features of the universe considered, i.e., of the e-mails of the SpamAssassin corpus. With fewer features available, the feature selection process produces even more significant features, further reducing the number of null patterns generated.

In spite of the greater reduction in the quantity of null patterns, and consequently, of the increase in the size of the set of training patterns, the average computational time spent on training is shorter because, by using more significant features in the training patterns, the training convergence is faster.

By analyzing the Figure 10, it may be noticed that the average computational times spent on testing, by the neural model, although very similar in the four experiments, present a downtrend, going from experiment 1 to experiment 4. This fact may be explained, once again, by the quantity of patterns in the test sets.

7 Experiments on the Ling-Spam corpus

The objective of experiment 5 is to evaluate the results in the classification of spam and ham e-mails of the Ling-Spam corpus, when using both the pre-processing techniques and the feature selection methods.

7.1 Experiment 5 — Method FD

Table 27 details the training, validation and test sets used in the experiment.

Table 27: Ham and spam patterns in the training, validation and test sets of experiment 5 — FD method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	963	192	0	0	0	771	1926
Validation	481	96	0	0	0	385	962
Test	963	191	5	2	0	772	1933

It may be noticed in Table 27 that the use both of pre-processing and of the FD method produces a small quantity of null patterns.

Tables 28 and 29 present, respectively, the results obtained with the use of the FD method in the experiment and the computational times spent on the training and testing of the neural model.

Table 28: Results of experiment 5 — FD method

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	96.69 ± 0.62	93.39 ± 0.54
10	97.14 ± 0.68	93.84 ± 1.09
15	97.86 ± 0.53	95.34 ± 1.82
20	98.63 ± 0.33	94.08 ± 0.90
25	98.88 ± 0.24	94.32 ± 1.46
30	98.92 ± 0.26	95.63 ± 1.16
35	98.68 ± 0.55	96.19 ± 1.98
40	99.11 ± 0.27	96.20 ± 0.85
45	98.99 ± 0.24	95.94 ± 1.37
50	99.26 ± 0.18	95.72 ± 1.31
55	99.28 ± 0.26	94.93 ± 1.60
60	99.12 ± 0.31	96.24 ± 1.10
65	99.17 ± 0.21	96.50 ± 0.84
70	99.10 ± 0.12	96.76 ± 1.21
75	99.35 ± 0.34	96.95 ± 1.29
80	*99.52* ± 0.12	96.22 ± 0.99
85	99.50 ± 0.17	97.10 ± 0.93
90	99.43 ± 0.21	*97.14* ± 0.87
95	99.50 ± 0.34	96.23 ± 0.87
100	99.48 ± 0.21	96.52 ± 1.39

Table 29: Computational times (in seconds) spent on the training and testing in experiment 5 — FD method

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	1.53	*0.04*
10	1.55	0.04
15	1.55	0.04
20	1.75	0.04
25	1.67	0.04
30	1.84	0.04
35	1.74	0.04
40	1.72	0.04
45	1.86	0.04
50	1.80	0.05
55	1.73	0.05
60	1.78	0.04
65	1.83	0.05
70	1.80	0.05
75	1.84	0.05
80	1.96	0.05
85	2.12	0.05
90	1.57	0.05
95	1.54	0.05
100	*1.46*	0.05

7.2 Experiment 5 — Method Chi-Square

Table 30 details the training, validation and test sets used in the experiment.

Table 30: Ham and spam patterns in the training, validation and test sets of experiment 5 — Chi-Square method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	928	192	0	0	0	736	1856
Validation	464	96	0	0	0	368	928
Test	927	193	93	0	0	734	1947

It may be noticed in the Table 30 that the use of pre-processing and of the Chi-Square method did not produce null spam patterns. On the other hand,

the quantity of null ham patterns grew significantly.

Tables 31 and 32 present, respectively, the results obtained with the use of the Chi-Square method in the experiment and the computational times spent on the training and testing of the neural model.

Table 31: Results of experiment 5 — Chi-Square method

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	98.63 ± 0.04	91.83 ± 0.10
10	94.94 ± 2.08	96.46 ± 1.59
15	98.01 ± 1.33	94.61 ± 1.44
20	98.34 ± 0.79	94.97 ± 1.29
25	98.90 ± 0.13	95.15 ± 0.48
30	98.68 ± 0.17	95.16 ± 0.47
35	98.86 ± 0.30	94.37 ± 1.40
40	99.03 ± 0.35	94.19 ± 0.68
45	98.91 ± 0.40	94.10 ± 1.05
50	98.69 ± 0.33	94.52 ± 1.05
55	98.67 ± 0.51	94.76 ± 1.13
60	98.69 ± 0.24	95.20 ± 1.21
65	98.73 ± 0.32	95.50 ± 2.28
70	98.74 ± 0.59	95.83 ± 1.75
75	98.53 ± 0.75	95.62 ± 1.24
80	98.82 ± 0.56	95.26 ± 2.09
85	98.77 ± 0.46	96.63 ± 1.10
90	98.72 ± 0.46	*96.64* ± 0.70
95	98.99 ± 0.33	96.29 ± 1.32
100	*99.04* ± 0.34	96.70 ± 0.94

Table 32: Computational times (in seconds) spent on the training and testing in experiment 5 — Chi-Square method

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	2.51	*0.04*
10	*1.46*	0.04
15	1.52	0.04
20	1.70	0.04
25	1.52	0.04
30	1.54	0.04
35	1.62	0.04
40	1.57	0.04
45	1.57	0.04
50	1.54	0.04
55	1.60	0.04
60	1.59	0.04
65	1.60	0.05
70	1.62	0.05
75	1.61	0.05
80	1.64	0.05
85	1.66	0.05
90	1.80	0.05
95	1.75	0.05
100	1.81	0.05

7.3 Experiment 5 — Method MI

Table 33 details the training, validation and test sets used in the experiment.

Table 33: Ham and spam patterns in the training, validation and test sets of experiment 5 — MI method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	952	192	0	0	0	760	1904
Validation	476	96	0	0	0	380	952
Test	951	192	33	1	0	759	1936

It may be observed in the Table 33 that just like in experiment 4, the MI method produces a higher quantity of null patterns than that produced by the FD method and lower than that produced by the Chi-Square method.

Tables 34 and 35 present, respectively, the results obtained with the use of the MI method in the experiment and the computational times spent on the training and testing of the neural model.

7.4 Analysis of the results obtained on the Ling-Spam corpus (experiment 5)

Figures 11 and 12 present the percentages of correct classifications of ham and spam e-mails obtained in experiment 5, according to the number of features selected in each of the three methods.

By analyzing Figures 11 and 12, it may be observed that both in the classification of ham and of spam e-mails, the total percentage of correct classifications grows with the increase in the dimensionality of the input pattern. In the classification of ham e-mails, however, the total percentage of correct classifications tends to stabilize after a particular dimensionality value.

In the classification of ham e-mails, the three feature selection methods presented similar performances. For spam e-mails, however, the better performance of the FD method is perceptible in the graph. This fact is very likely due to the fact that the Ling-Spam corpus originates from few sources of information, thus accentuating the distance between the ham and spam classes. This feature of the Ling-Spam corpus favours selection by the FD method, as this does not take into account the relevance of a feature to the two classes — ham and spam.

Generally speaking, the results obtained on the Ling-Spam corpus appear superior to those obtained on the SpamAssassin corpus. This can be explained

Table 34: Results of experiment 5 — MI method

No. Inputs	Correct Classification (%)	
	Ham	Spam
5	88.02 ± 1.92	92.23 ± 1.01
10	97.24 ± 0.43	92.06 ± 0.68
15	97.52 ± 0.78	92.49 ± 0.90
20	97.80 ± 0.34	92.48 ± 0.93
25	97.99 ± 0.83	94.26 ± 0.89
30	98.14 ± 0.52	94.09 ± 1.58
35	98.74 ± 0.33	93.88 ± 0.96
40	98.85 ± 0.38	94.21 ± 1.29
45	98.76 ± 0.57	94.67 ± 1.80
50	98.72 ± 0.58	94.88 ± 1.30
55	*99.08* ± 0.30	93.74 ± 1.39
60	98.42 ± 0.75	95.24 ± 1.15
65	98.84 ± 0.39	95.02 ± 1.42
70	98.65 ± 0.73	95.97 ± 1.97
75	98.78 ± 0.49	95.97 ± 1.45
80	98.54 ± 0.53	96.52 ± 0.81
85	99.07 ± 0.46	95.78 ± 1.36
90	98.96 ± 0.47	96.18 ± 1.29
95	98.88 ± 0.50	96.52 ± 1.21
100	98.65 ± 0.84	*96.63* ± 1.10

Table 35: Computational times (in seconds) spent on the training and testing in experiment 5 — MI method

No. Inputs	Mean Time (sec.)	
	Training	Testing
5	1.87	0.05
10	1.97	0.05
15	1.83	*0.04*
20	1.82	0.05
25	1.87	0.06
30	*1.57*	0.04
35	1.68	0.04
40	1.71	0.04
45	1.63	0.04
50	1.67	0.04
55	1.90	0.05
60	1.66	0.05
65	1.78	0.05
70	1.66	0.05
75	1.73	0.05
80	1.70	0.05
85	1.83	0.05
90	1.58	0.05
95	1.66	0.05
100	1.66	0.05

by the fact that the Ling-Spam corpus is smaller and more uniform as regards the sources of information, making the classification work of the neural model easier.

Finally, by analyzing Tables 29, 32 and 35 of experiment 5, it may be observed that the average computational times spent on training, by the neural model, ranged from 1.46 to 2.51 seconds. The Tables show that for input patterns containing from ten to eighty-five features, the shorter average training times were reached with the use of the Chi-Square method. However, with this method, the neural model produced the worst performance outside this region. The average computational times spent on testing, by the neural model, present performance as expected. The time spent grows with the number of features present in the input pattern.

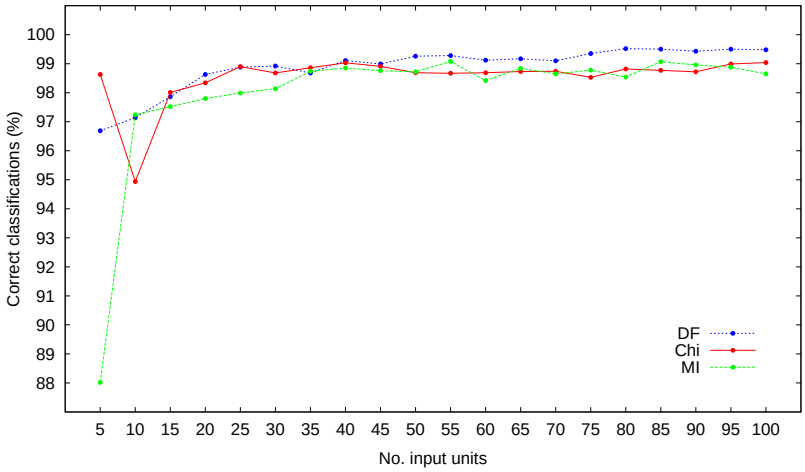

Figure 11: Comparison of the three methods in the classification of ham e-mails in experiment 5

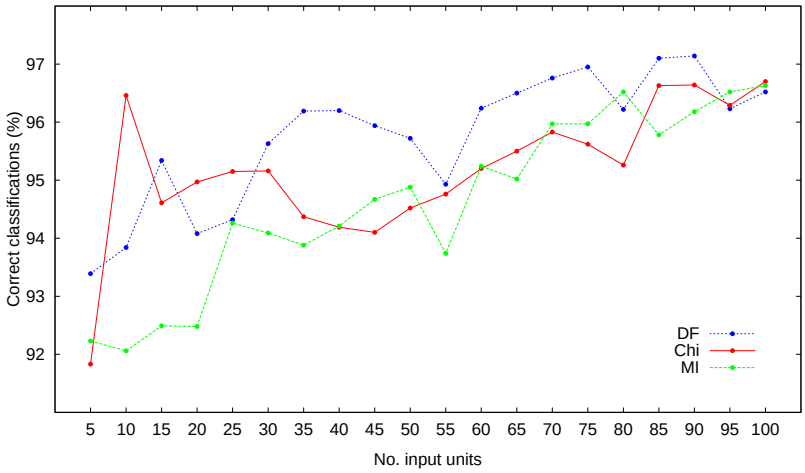

Figure 12: Comparison of the three methods in the classification of spam e-mails in experiment 5

8 Experiments on the Trec 2007 corpus

The experiments involving the Trec 2007 corpus also used both pre-processing techniques and feature selection methods. The Trec corpus originated from

Spam Track, an event of the annual Text Retrieval Conference (TREC) whose scope is research into text recovery.

The Trec 2007 corpus contains an extensive collection of e-mails, including e-mails with images. For this reason, it has already been employed in studies involving classification of e-mails with images [6]. However, no specific treatment for the images was performed, maintaining the same pre-processing techniques employed in the previous experiments.

Owing to the good results from the experiments with SpamAssassin and Ling-Spam corpora, the experiments with the Trec corpus started with patterns of ten features, varying every ten features until it reached one hundred features.

8.1 Experiment 6 — Method FD

Table 36 details the training, validation and test sets used in the experiment.

Table 36: Ham and spam patterns in the training, validation and test sets of experiment 6 — FD method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	10021	19841	0	0	9820	0	39682
Validation	5011	9920	0	0	4909	0	19840
Test	10021	19841	161	472	9820	0	40315

It may be seen in the Table 36 that although there be a high quantity of null patterns, in percentage terms, this quantity is below 1% of the total quantity of e-mails of the Trec corpus.

Tables 37 and 38 present, respectively, the results obtained with the use of the FD method in the experiment and the computational times spent on the training and testing of the neural model.

8.2 Experiment 6 — Method Chi-Square

Table 39 details the training, validation and test sets used in the experiment.

It may be seen in the Table 39 that the Chi-Square method, in relation to the FD method, presented a lower quantity of null ham patterns and, on the other hand, a much higher quantity of null spam patterns.

Tables 40 and 41 present, respectively, the results obtained with the use of

Table 37: Results of experiment 6 — FD method

No. Inputs	Correct Classification (%)	
	Ham	Spam
10	94.08 ± 1.44	90.88 ± 0.70
20	96.29 ± 0.66	92.04 ± 0.91
30	97.85 ± 0.23	93.54 ± 1.09
40	97.87 ± 0.23	93.90 ± 0.66
50	97.83 ± 0.19	94.92 ± 0.55
60	98.02 ± 0.40	95.64 ± 0.84
70	98.00 ± 0.37	95.39 ± 0.47
80	98.17 ± 0.26	96.26 ± 1.12
90	*98.34* ± 0.27	96.05 ± 0.88
100	98.21 ± 0.16	*96.65* ± 0.53

Table 38: Computational times (in seconds) spent on the training and testing in experiment 6 — FD method

No. Inputs	Mean Time (sec.)	
	Training	Testing
10	147.78	*0.43*
20	168.51	0.46
30	174.02	0.48
40	172.39	0.53
50	177.06	0.55
60	152.27	0.61
70	144.46	0.64
80	143.69	0.66
90	142.76	0.69
100	*130.95*	0.74

Table 39: Ham and spam patterns in the training, validation and test sets of experiment 6 — Chi-Square method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	10036	19692	0	0	9656	0	39384
Validation	5018	9846	0	0	4828	0	19692
Test	10037	19692	123	844	9655	0	40351

the Chi-Square method in the experiment and the computational times spent on the training and testing of the neural model.

8.3 Experiment 6 — Method MI

Table 42 details the training, validation and test sets used in the experiment.

As presented by Table 42, the quantity of null ham patterns is higher than that produced by the two previous methods. However, the quantity of null spam patterns lies between the quantity produced by the FD method and that produced by the Chi-Square method.

Tables 43 and 44 present, respectively, the results obtained with the use of the MI method in the experiment and the computational times spent on the training and testing of the neural model.

Table 40: Results of experiment 6 — Chi-Square method

No. Inputs	Correct Classification (%)	
	Ham	Spam
10	90.53 ± 0.66	95.49 ± 0.31
20	96.40 ± 0.34	96.58 ± 1.00
30	97.59 ± 0.15	98.51 ± 0.22
40	97.97 ± 0.20	98.53 ± 0.36
50	98.06 ± 0.21	*98.62* ± 0.40
60	98.08 ± 0.14	98.51 ± 0.32
70	*98.29* ± 0.15	98.59 ± 0.34
80	97.72 ± 0.70	98.38 ± 0.30
90	97.32 ± 1.21	98.45 ± 0.37
100	97.42 ± 0.79	98.52 ± 0.46

Table 41: Computational times (in seconds) spent on the training and testing in experiment 6 — Chi-Square method

No. Inputs	Mean Time (sec.)	
	Training	Testing
10	208.08	*0.42*
20	251.77	0.45
30	180.48	0.48
40	153.82	0.52
50	128.63	0.55
60	125.24	0.59
70	116.27	0.62
80	99.48	0.65
90	*87.63*	0.68
100	94.80	0.73

Table 42: Ham and spam patterns in the training, validation and test sets of experiment 6 — MI method

Sets	Valid Patterns		Null Patterns		Duplicated Pat.		Total
	Ham	Spam	Ham	Spam	Ham	Spam	
Training	9995	19769	0	0	9774	0	39538
Validation	4997	9884	0	0	4887	0	19768
Test	9995	19770	227	651	9775	0	40418

8.4 Analysis of the results obtained on the Trec corpus (experiment 6)

Figures 13 and 14 present the percentages of correct classifications of ham and spam e-mails obtained in experiment 6, according to the number of features selected in each of the three methods.

By analyzing Figures 13 and 14, it may be observed that, as was the case with the SpamAssassin corpus, the total percentage of correct classifications tends to stabilize after a particular value of dimensionality of the input pattern.

The Figures also indicate that no feature selection method was superior to the others in all the situations. In the classification of ham e-mails, the method of best performance, on average, was the FD, yet it was outperformed by the Chi-Square method in a few input pattern dimensions.

Table 43: Results of experiment 6 — MI method

No. Inputs	Correct Classification (%)	
	Ham	Spam
10	93.22 ± 1.46	83.62 ± 1.14
20	94.92 ± 0.47	93.97 ± 0.45
30	95.83 ± 0.49	94.71 ± 0.46
40	96.68 ± 0.54	95.45 ± 0.58
50	96.93 ± 0.13	96.06 ± 0.47
60	97.22 ± 0.18	96.54 ± 0.67
70	97.03 ± 0.33	97.17 ± 0.53
80	97.28 ± 0.85	97.35 ± 0.91
90	97.81 ± 0.27	97.85 ± 0.72
100	*97.88* ± 0.35	*98.23* ± 0.38

Table 44: Computational times (in seconds) spent on the training and testing in experiment 6 — MI method

No. Inputs	Mean Time (sec.)	
	Training	Testing
10	163.29	*0.43*
20	223.73	0.46
30	174.80	0.50
40	162.55	0.55
50	169.80	0.59
60	184.33	0.63
70	149.05	0.66
80	*129.53*	0.70
90	139.97	0.73
100	134.87	0.77

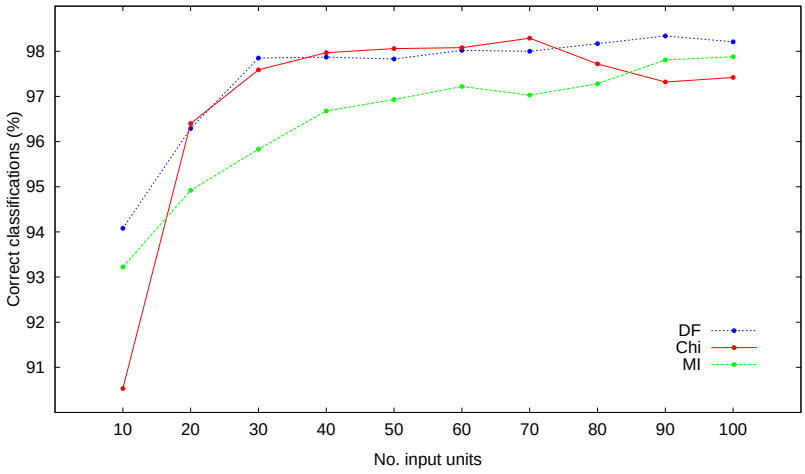

Figure 13: Comparison of the three methods in the classification of ham e-mails in experiment 6

In the classification of spam e-mails, the method with the best performance was indisputably the Chi-Square method which, given its good performance in the classification of ham e-mails as well, may be considered the one that achieved the best overall performance on the Trec corpus. On the other hand, the MI method, which presented satisfactory performance on the SpamAssassin and Ling-Spam corpora, did not achieve the same performance on the Trec corpus.

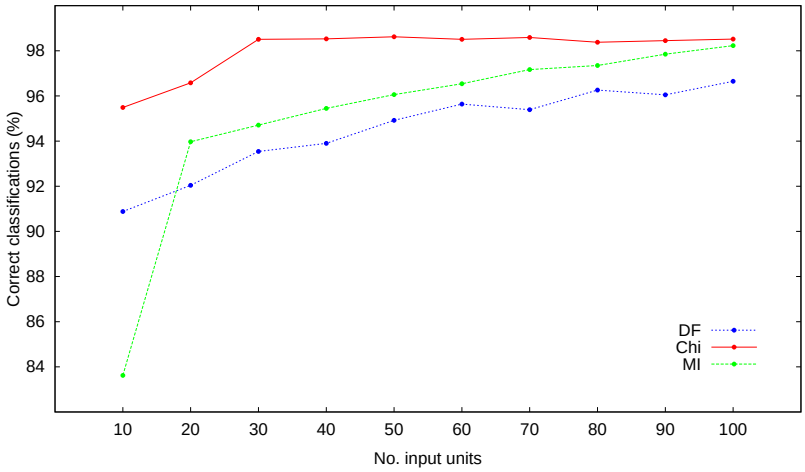

Figure 14: Comparison of the three methods in the classification of spam e-mails in experiment 6

As observed in the Figures and Tables of experiment 6, even though the Trec corpus presents a larger quantity of e-mails with diverse formats (including images), the neural model was able to satisfactorily identify the occurrence of ham and spam e-mails, with correct classification percentages above 98%.

Finally, by analyzing Tables 38, 41 and 44 of experiment 6, it may be observed that the average computational times spent on training, by the neural model, ranged from 87 to 252 seconds. The Tables show that for input patterns containing more than thirty features, the shortest average training times were reached with the use of the Chi-Square method. The average computational times spent on the tests, by the neural model, present expected behaviour. The times grow with the increase in dimensionality of the input vector.

9 A comparison of the three feature selection methods

As presented in experiments 4 to 6, none of the three feature selection methods employed was the best in all the situations. Such a fact results mainly from the peculiarities of the sources of information used for formation of the corpora — SpamAssassin, Ling-Spam and Trec 2007 — used.

Table 45 presents the average performance of the neural model with the use of each of the three methods on the three corpora. The average performance is

measured through the simple mean of the average percentage of correct classifications, of all dimensions of the input pattern, obtained by the neural model with the use of the method for a particular class — ham and spam — on the corpus evaluated.

Table 45: Performance of the neural model with the use of the three feature selection methods

Meth.	SpamAssassin corpus			Ling-Spam corpus			Trec 2007 corpus		
	Ham	Spam	Acc.	Ham	Spam	Acc.	Ham	Spam	Acc.
FD	95.70	93.93	94.99	*98.88*	*95.76*	*97.63*	*97.47*	94.53	96.29
Chi	*97.49*	94.37	96.24	98.53	95.19	97.19	96.94	*98.02*	*97.37*
MI	97.38	*94.81*	*96.35*	97.98	94.64	96.64	96.48	95.10	95.93

Table 45 also presents an indicator of accuracy, calculated from the equation

$$Accuracy = 0.6 \cdot (HamMean) + 0.4 \cdot (SpamMean) \tag{5}$$

where *HamMean* and *SpamMean* are, respectively, the simple means of the average percentage of correct classifications on ham and spam e-mails. A higher weight is assigned to correct classifications of ham e-mails owing to the problem of false positives, i.e., incorrect classifications of ham e-mails are recognized as more costly than incorrect classifications of spam e-mails.

In Table 45, it may also be seen that the MI method presented the best performance of the three methods, on the SpamAssassin corpus. However, the MI method achieved an inferior performance to the other two methods on the Ling-Spam and Trec 2007 corpora, in which the methods of best performance were FD and Chi-Square, respectively.

No feature selection method was superior to the others in all the situations among the cases studied. Yet, they all performed well in representing the features of the e-mails, helping the neural model to achieve classification rates equal to or above 95%.

10 Experiment 7 — obfuscation

Cournane and Hunt [11] describe a series of techniques used by *spammers* in the attempt to deceive anti-spam systems, such as the use of invalid HTML tags, use of invisible text, use of blank HTML tags, use of HTML comments, and others. The anti-spam prototype does not implement procedures to deal with the majority of these obfuscation techniques. The development of procedures to deal with them is important and will be addressed in future research.

However, the anti-spam prototype does implement procedures to deal with one of the techniques mentioned by Cournane and Hunt [11], which is the technique which aims to conceal words with invalid characters. Experiment 7 was conducted with the intention of verifying the performance of the anti-spam prototype in the presence of e-mails which make use of this obfuscation technique.

Experiment 7 used the MI method for selection of features, and the same data as experiment 4. After this, ten ham e-mails were chosen randomly. A word with invalid characters, such as 'V I A G R A, V*I@GR@, and others, was inserted in each of these ten e-mails.

The neural model was trained with a training set that included the ten altered e-mails instead of the ten original ones. For statistical purposes, the neural model training process was performed ten times for each dimension of the input pattern.

Figure 15 presents the average percentage of correct classifications, as spam, of these ten e-mails, as well as the variation of this average percentage according to the dimensionality of the input pattern.

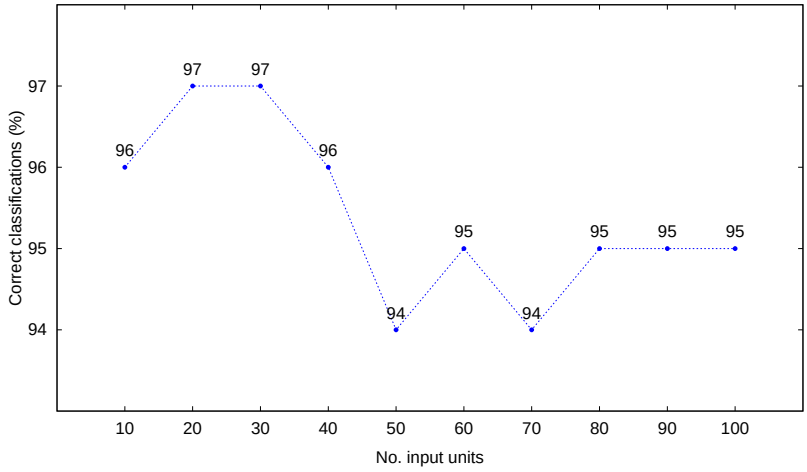

Figure 15: Average percentages of correct classifications of the ten e-mails

From the results presented in Figure 15, it may be verified that the vast majority of these ten e-mails was classified correctly as spam. As the dimensionality of the input pattern increases, the average percentage of correct classifications presents a slight reduction. This is due to the fact that the input patterns, with the increase in dimensionality, begin to include more features relating to ham e-mails, for these ten e-mails were originally ham e-mails.

11 Related work

Zorkadis et al. [37] make use of the Ling-Spam corpus in a study of models to detect spam e-mails, aiming to reduce the number of false positives and negatives. The corpus is processed in such a way as to reduce the words to their radical forms. Moreover, based on a list, words of minor significance, such as *a*, *as*, *the*, *for* and others, are eliminated. The Ling-Spam corpus was divided into two sets — one a training and the other a test set, with 60% and 40% of the e-mails, respectively. The 100 and 500 most significant features were selected by the MI method. The authors used the Naïve-Bayesian, AdaBoostM1, Classification via Regression, MultiBoostAB, Random Committee, ADTree, ID3-Tree and RandomTree models as well as the model proposed by the authors, which combines the Random Committee and ADtree methods, complementary with regard efficiency in the classification of false positives and false negatives.

The authors conclude that the model with the lowest incidence of false positives was Random Committee, whilst ADtree and ID3Tree were those with the best performance in relation to false negatives. The authors do not present percentages of correct classifications of the e-mails. However, they mention that starting from 500 features, the proposed models incorrectly classify between 1 and 5 ham e-mails and between 11 and 52 spam e-mails.

Chuan et al. [9] compare the performance of three models — multilayer perceptron (MLP), least vector quantization (LVQ) and Naïve-Bayesian — for classification of e-mails. They use the SpamAssassin corpus, from where 580 spam e-mails and 420 ham e-mails are randomly selected, and the MI method, for selection of the 100 most relevant features of the e-mails.

They propose a method composed of two stages. The first serves to define the subclass to which the e-mail belongs. The e-mails are then divided into subclasses, according to the category — promotions, shopping, adult content, and others — into which they fall. In this stage, the three models are trained to classify the e-mails in these subclasses. In the second stage, the models are trained to classify the e-mails from these subclasses in only two classes — ham and spam. The results obtained by the authors indicate that the LVQ model presents a performance superior to that of the MLP, with percentages of correct classifications of 98% and 93% for the spam precision and spam recall indices, respectively. The MLP model obtains results of 90% and 91% for these indices. The results obtained by the Naïve-Bayesian model are inferior to those obtained by the other two models.

Xu and Yu [32] propose the use of a lexical analyzer, which correlates the occurrence of a word with the others. They also propose a change in the back-propagation training algorithm of a multilayer perceptron (MLP), in order to optimize the training time and to avoid local minima. The paper used the Ling-Spam corpus, with the substitution of the words by their radical forms. One

thousand e-mails, selected randomly, were used to train and test the MLP. The e-mails were divided into ten parts, with nine used in the training and validation and one part used in the tests. With this high imbalance between the training and test sets, the MLP, employing the reviewed backpropagation together with the lexical analyze, obtained the best results, with accuracy above 98%. The authors conclude that the reviewed backpropagation produces the greatest impact on the quality of results, but that the use of the lexical analyzer also contributes to this quality.

Park and Deshpande [25] propose the use of a hybrid system composed of three techniques — detection lists, content verification and detection of forged e-mails — for the detection of spam e-mails. White and black lists are the detection lists used most often. Senders of legitimate e-mails (ham) are registered in the whitelists, whilst senders of spam e-mails are registered in the blacklists. Both lists may be implemented both at the e-mail client and at the e-mail server.

The techniques based on content verification generally analyze the body and attachments of the e-mail in search of key features that indicate the legitimate and non-legitimate nature of the message. Such techniques are basically supported by the categorization of texts and structures, making use of statistical and neural models for the classification of e-mails.

The detection of forged e-mails is performed by means of the identification of false information inserted in the header fields of the e-mails. The techniques for insertion of false information can be very sophisticated nowadays, sometimes requiring the system to track the source IP address of the e-mail.

The authors employ the techniques in the following order. The integrity of the domain name (DNS) of the e-mail is checked first. Then it is verified whether the source address of the e-mail is present in a blacklist. Thirdly, the *From* field is compared with the domain information obtained via DNS. In fourth place, a search is performed in a whitelist to verify whether the source address is present, and finally, the e-mail content is analyzed.

The corpus used is composed of 905 e-mails, of which 234 are ham and 671 spam. The corpus was formed by e-mails collected from the mailbox of a single user. The authors compare the results obtained by the hybrid method with the individual use of each of the techniques. The hybrid system presented a superior performance in relation to each one of the techniques applied individually, reaching percentages of 98% for the spam and ham precision indices and percentages of 99% and 96%, respectively, for the spam and ham recall indices. Finally, the authors address the limitation of having used e-mails of a single user to build the corpus. They intend to extend their studies to other corpora in the future in order to observe the performance of the proposed system in an environment closer to the reality found by anti-spam systems.

For the detection of spam e-mails, Kim et al. [17] propose the use of fuzzy

logic to select the features of a corpus of e-mails. The authors use a support vector machine (SVM) as an e-mail classification model. The corpus used is composed of 4792 e-mails, of which 2218 are ham, 1100 spam of pornographic content, 1077 spam of financial content and 397 spam of commercial content. The authors compare two feature selection methods — Information Gain and Chi-Square — with their approach, which uses fuzzy logic. They conclude that the use of fuzzy logic to select features of the e-mails induces the SVM to produce percentage results of 79% and 91%, respectively, for the spam precision and spam recall indices. With the use of the Information Gain and Chi-Square methods, SVM produces results, on average, between 6% to 10% lower in these indices.

Wang and Chen [31] propose the identification of ham and spam e-mails through the analysis of their headers. They suggest that the identification of suspicious headers can be performed through the verification of the occurrence of the main techniques used by spammers to conceal the origin of e-mails. For example, they mention that only 7.2% of spam e-mails have a destination address in the *To* and *CC* fields. Therefore, most of them have such an address in the *blind carbon copy (BCC)* field. Furthermore, the *X-Mailer* field, which indicates the client software used to send the e-mail (MUA), generally presents distorted information, as many spam e-mails are originated from softwares that automate the sending of messages. Likewise, the *message identifier (Message-ID)* field, composed of two parts separated by the symbol @, is altered with the intention of concealing the real name of the domain of origin of the spam e-mail, or of the first hop through which it passed. Hence the domain presented in the field of the sender does not match the domain presented in *Message-ID*.

The experiments were carried out on a corpus containing 10024 spam e-mails, collected from the SpamArchive corpus and 1234 ham e-mails, collected by three volunteers over a one-week period. Based on the analysis of the features of the collected e-mails, the authors determine three rules to consider an e-mail as ham, and four rules to consider it spam. The authors conclude that if the user utilizes a conservative strategy of only blocking e-mails that fully comply with the rules, these do not generate false positives. However, the number of false negatives produced is around 20.89%. If an aggressive strategy is used, the rules are able to identify 92.5% of the spam e-mails. In this case, however, the rate of false positives rises to 10.28%.

Byun et al. [5] employ the Trec 2005 and Trec 2007 corpora. They propose the use of an anti-spam system made up of a specific Bayesian model to classify e-mails which included images and of another Bayesian model to classify all the other e-mails. Although the achieved results indicate high percentages of correct classifications on ham e-mails on both the Trec 2005 and Trec 2007 corpora, the same is not achieved in relation to the percentage on spam e-mails. For example, the best false negative rate was 6.42% on Trec 2005 corpus, which indicates that their model classified correctly 93.60% of the spam e-mails in this corpus.

Amayri and Bouguila [1] assess the performance of support vector machines in spam filtering when using different kernel functions. On the Trec 2005 corpus, they obtained from 71% to 91% of accuracy in e-mail classifications with classical distance-based kernels whereas with string kernels this percentage varied from 89% to 93%. However, the computational time spent by the SVMs with string kernels to classify the e-mails was one order of magnitude higher than that spent by the SVMs with classical kernels.

Behjat et al. [3] use a multilayer perceptron to classify e-mails. The features of the e-mails are selected by a genetic algorithm. The training and testing sets contain respectively 90% and 10% of the Ling-Spam corpus. With this high imbalance between the two sets, the MLP was capable of classifying correctly 99.68% of the e-mails from the testing set.

Caruana et al. [8] propose a parallel SVM training algorithm based on the MapReduce technique. The training data is divided into several subsets, and each one of these subsets is then used to train one SVM classifier which run in a computer node. To avoid accuracy degradation when splitting the training data among several SVM classifiers, they propose the use of ontology-based semantics. The authors comproved two facts from the experiments. Firstly, that the training computational time with the proposed algorithm was significantly faster than that with the sequential SVM training algorithm. Secondly, that the use of ontology-based semantics improved slightly the accuracy of the parallel SVM on both the Spam-Base and Trec 2007 corpora.

El-Alfy and Abdel-Aal [13] propose an abductive network ensemble to classify e-mails. The ensemble was trained with 62% of the Spam-Base corpus, and tested with the remainder of the corpus. It was compared with neural and Naïve-Bayesian classifiers. The abductive network ensemble presented 92.4% of accuracy on the testing set whilst the neural and Naïve-Bayesian classifiers presented 91.7% and 75.4%, respectively.

Gong and Chen [14] test a Bayesian anti-spam system. The system architecture includes an user feedback module. Selection of e-mail features was based on information gain. Selected features varied from 50 to 350. The system was trained and tested on e-mails from the Ling-Spam corpus. The results could achieve a maximum accuracy of 97.68% with the use of 350 features.

Li and Huang [18] propose an e-mail classifier which integrates three kinds of semantic similarity approaches to select e-mail features, and employs a multilayer perceptron to classify the e-mails. They make use of the Ling-Spam, PU1, and PU3 corpora to train and test the classifier. The training to testing set ratio was 90% to 10%. With this high ratio, the classifier was capable of classifying correctly up to 98% of the e-mails from the three corpora.

Marsono et al. [19] propose a spam detection technique at the datagram level. In such technique, the e-mails are pre-classified in transit on a per-datagram

basis by using an adapted Bayesian classifier. The results of their technique were, depending on the datagram size, from 3% to 11% inferior to that of an usual Bayesian filter acting in the entire e-mail.

Nosseir et al. [21] test a word-based anti-spam system. In the pre-processing phase, words such as articles, prepositions, and conjunctions are removed from the e-mails, misspelled words are corrected, and derived words are converted to their root forms. The remaining words are then classified into two classes — good or bad. Bad words are those which appeared previously in spam e-mails. An ensemble of multilayer perceptrons identifies the good and bad words in the e-mails, and a decision function classifies the e-mails as ham or spam. In a particular set of 120 words, their anti-spam system was able to classify correctly 93% and 95% of the ham and spam e-mails, respectively.

Panigrahi [23] compares the performance of eleven machine learning models to classify e-mails extracted from the UCI machine learning repository. The three best models were the neural network, the support vector machine, and C5.0, with, respectively, 93.86%, 92.81%, and 92.01% of accuracy. The three worst models were the random forest, J48, and lazy-LWL, with 80.96%, 80.96%, and 69.84% of accuracy, respectively.

Renuka et al. [27] also compare the performance of machine learning models to classify e-mails from the UCI machine learning repository. The models tested are the J48, the MLP, and the Naïve-Bayesian with the original and the filtered Bayesian learning algorithms. The training to testing set ratio was 90% to 10%. With this high ratio, the J48, the MLP, and the Naïve-Bayesian models achieved, respectively, 92%, 93%, and 89% of accuracy. With the filtered Bayesian learning algorithm, the Naïve-Bayesian model improved its accuracy to 91%.

Su et al. [28] propose a neural tree to e-mail classification. According to the authors, the neural tree incorporates the advantages of both decision trees and neural networks. They test the neural tree on three corpora — SpamAssassin, SpamTrack, and Trec 2006 Chinese. The authors select 38 features from the header parts of the e-mails. The neural tree is compared to the Naïve-Bayesian, the sequential minimal optimization, and C4.5 models. The neural tree performed better than the other three models, achieving 89.15%, 90.87%, and 99.85% of accuracy on the SpamAssassin, SpamTrack, and Trec 2006 Chinese corpora, respectively.

Ting and Qingsong [30] propose two methods — mutual information method with frequency (MIf) and mutual information method with average frequency (MIaf) — to select features from e-mails. The MIf and MIaf methods introduce, respectively, a word frequency factor and a word average frequency factor to the original mutual information method. They use the Naïve-Bayesian model to classify e-mails from the PU1 and CCERT corpora. The results showed that the Naïve-Bayesian model performed better when the e-mail features were

selected by the MIf and MIaf methods.

Ying et al. [34] propose an ensemble model to classify e-mails. The ensemble model is made up of the decision tree, support vector machine, and multilayer perceptron models. The training and test sets consist of e-mails colected from the mail server of Huafan University and from the Trec 2007 corpus. They verified that the accuracy on e-mail classification of the ensemble model was better than that of each one of the three models, individually.

Zhou et al. [36] propose a three-way model to classify e-mails. In the three-way model, the e-mails are classified as ham, spam, or indeterminate. The model also makes use of a loss function which expresses the costs of making classification decisions. The model is trained and tested on three e-mail corpora — PU1, Ling-Spam, and UCI machine learning repository —, and is compared to other two ternary anti-spam models and to the Naïve-Bayesian model. The three-way model achieved the best performance in terms of cost.

12 Conclusion

Spam e-mails, disseminated by individuals known as spammers, constitute a serious problem nowadays, as they affect both the functioning of the Internet and its users. The volume of spam e-mails circulating on the Internet produces serious inconveniences, such as mailboxes filled with undesirable e-mails, expenditures with the traffic of undesirable e-mails, time spent on the elimination of these e-mails, as well as serious financial losses for corporations and for users themselves.

The report proposes a novel three-stage anti-spam (AS) prototype. The first stage performs an e-mail pre-processing in which the e-mails are converted into a more uniform form, their irrelevant contents are removed, and some known spam patterns are detected. The second, through feature selection methods, identifies the main features of the e-mails. The third consists of a neural model to classify the e-mails into two classes — ham and spam.

The AS prototype was thoroughly tested on three public corpora — SpamAssassin, Ling-Spam and Trec 2007 —, in a series of experiments. The experiments employed three statistical feature selection methods — frequency distribution (FD), chi-square (Chi) and mutual information (MI) —, widely used in classification of texts, in the area of machine learning. A multilayer perceptron (MLP) was used as a neural model on account of its simplicity, speed in training and classification, and recognized generalization ability.

The experiments also aimed to measure the contribution of each one of the three stages in the end result of classification of the e-mails. Accordingly, based on the results obtained, it was proven that both the pre-processing stage and

that of feature selection are important for accuracy in e-mail classifications.

Among the three statistical feature selection methods employed, the MI method presented the best performance on the SpamAssassin corpus. However, it was outperformed by the FD method in the classification of e-mails of the Ling-Spam corpus, and by the Chi and FD methods in the classification of e-mails of the Trec corpus. These results suggest the fact that none of the methods employed is the best in all the situations, thus opening up a new field for future research.

The MLP neural model was employed with twenty different quantities of neurons in its input layer on the SpamAssassin and Ling-Spam corpora, and with ten different quantities of neurons on the Trec corpus, according to the dimensionality of the input pattern used. Although the fact that larger dimensionalities of the input pattern increase the accuracy of classifications be intuitive, it was observed that from a certain dimensionality value, the gain obtained tends to be very small.

Another important contribution of this report was that of measuring the computational times spent on the training and testing of the neural model. Once again, although the fact that greater dimensionalities of the input pattern require longer training and testing times be intuitive, it was observed that the use of input patterns with greater dimensionalities, yet containing the more relevant features selected by the feature selection methods, improve the convergence and the quality of training, producing shorter training times.

The results obtained by the AS prototype on the three corpora are promising. The processing times required for the pre-processing, selection of features, training and testing of the neural model are hardly significant at all, which allows the transformation of the AS prototype developed into a tool for corporate use.

Finally, it is important to mention that the AS prototype was entirely developed on a modularized, object-oriented architecture. Hence its architecture facilitates the addition of new components such as the handling of new spam patterns, for instance. This fact is important, as AS systems must be able to evolve in their classification techniques, in line with the evolution of techniques used by spammers.

For the continuation of this research, we may enumerate some directions for future work. Firstly, the evaluation of new statistical feature selection methods. Secondly, the development of a new module, in the AS prototype, to recognize other cases of obfuscation used by spammers, which are ignored by the current AS systems. Thirdly, the development of another new module, in the AS prototype, to identify image spam e-mails. In fourth place, the evaluation of other models, such as support vector machines, for the classification of e-mails. Finally, the transformation of the AS prototype into a technological tool, under

free software license, for corporate use.

Acknowledgements

This research was fully supported by CAPES, Brazil.

References

[1] O. Amayri and N. Bouguila. A study of spam filtering using support vector machines. *Artificial Intelligence Review*, 34:73–108, 2010.

[2] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naïve Bayesian anti-spam filtering. In *Proceedings of the Workshop on Machine Learning in the New Information Age*, pages 9–17, 2000.

[3] A. R. Behjat, A. Mustapha, H. Nezamabadi-Pour, N. Sulaiman, and N. Mustapha. GA-based feature subset selection in a spam/non-spam detection system. In *Proceedings of the International Conference on Computer and Communication Engineering (ICCCE)*, pages 675–679, 2012.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[5] B. Byun, C. Lee, S. Webb, D. Irani, and C. Pu. An anti-spam filter combination framework for text-and-image emails through incremental learning. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, July 2009.

[6] B. Byun, C. Lee, S. Webb, and C. Pu. A discriminative classifier learning approach to image modeling and spam image identification. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, 2007.

[7] G. Caruana and M. Li. A survey of emerging approaches to spam filtering. *ACM Computing Surveys*, 44(2):9:1–9:27, 2012.

[8] G. Caruana, M. Li, and Y. Liu. An ontology enhanced parallel SVM for scalable spam filter training. *Neurocomputing*, 108:45–57, 2013.

[9] Z. Chuan, L. Xianliang, H. Mengshu, and Z. Xu. A LVQ-based neural network anti-spam email approach. *ACM SIGOPS Operating Systems Review*, 39(1):34–39, 2005.

[10] G. V. Cormack and T. R. Lynam. 2007 TREC public spam corpus, 2011. available at `http://plg.uwaterloo.ca/~gvcormac/treccorpus07/`.

[11] A. Cournane and R. Hunt. An analysis of the tools used for the generation and prevention of spam. *Computers & Security*, 23:154–166, 2004.

[12] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[13] E. M. El-Alfy and R. E. Abdel-Aal. Using GMDH-based networks for improved spam detection and email feature analysis. *Applied Soft Computing*, 11:477–488, 2011.

[14] Y. Gong and Q. Chen. Research of spam filtering based on Bayesian algorithm. In *Proceedings of the International Conference on Computer Application and System Modeling (ICCASM)*, volume 4, pages 678–680, 2010.

[15] T. S. Guzella and W. M. Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36:10206–10222, 2009.

[16] S. Haykin. *Neural Networks and Learning Machines*. Prentice-Hall, Inc., third edition, 2009.

[17] J. Kim, S. Kang, and B. M. Kim. A fuzzy inference method for spam-mail filtering. In *Australian Joint Conference on Advances in Artificial Intelligence*, volume 3809 of *Lecture Notes in Artificial Intelligence*, pages 1112–1115, 2005.

[18] C. H. Li and J. X. Huang. Spam filtering using semantic similarity approach and adaptive BPNN. *Neurocomputing*, 92:88–97, 2012.

[19] M. N. Marsono, M. W. El-Kharashi, and F. Gebali. Targeting spam control on middleboxes: Spam detection based on layer-3 e-mail content classification. *Computer Networks*, 53(6):835–848, 2009.

[20] Natural Language Processing Group. *The Ling-Spam Corpus*, 2011. available at `http://nlp.cs.aueb.gr/software.html`.

[21] A. Nosseir, K. Nagati, and I. Taj-Eddin. Intelligent word-based spam filter detection using multi-neural networks. *International Journal of Computer Science Issues*, 10(2):17–21, 2013.

[22] L. Özgür, T. Güngör, and F. Gürgen. Adaptive anti-spam filtering for agglutinative languages: A special case for Turkish. *Pattern Recognition Letters*, 25:1819–1831, 2004.

[23] P. K. Panigrahi. A comparative study of supervised machine learning techniques for spam e-mail filtering. In *Proceedings of the International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 506–512, 2012.

[24] A. Papoulis and S. U. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 4 edition, 2001.

[25] J. S. Park and A. Deshpande. Spam detection: Increasing accuracy with a hybrid solution. *Information Systems Management*, 23(1):57–67, 2006.

[26] M. K. Paswan, P. S. Bala, and G. Aghill. Spam filtering: Comparative analysis of filtering techniques. In *Proceedings of the International Conference on Advances in Engineering, Science and Management (ICAESM)*, pages 170–176, 2012.

[27] D. K. Renuka, T. Hamsapriya, M. R. Chakkaravarthi, and P. L. Surya. Spam classification based on supervised learning using machine learning techniques. In *Proceedings of the International Conference on Process Automation, Control and Computing (PACC)*, pages 1–7, 2011.

[28] M. Su, H. Lo, and F. Hsu. A neural tree and its application to spam e-mail detection. *Expert Systems with Applications*, 37:7976–7985, 2010.

[29] The Apache Software Foundation. *SpamAssassin Project*, 2011. available at `http://spamassassin.apache.org/publiccorpus/`.

[30] L. Ting and Y. Qingsong. Spam feature selection based on the improved mutual information algorithm. In *Proceedings of the International Conference on Multimedia Information Networking and Security (MINES)*, pages 67–70, 2012.

[31] C. Wang and S. Chen. Using header session messages to anti-spamming. *Computers & Security*, 26:381–390, 2007.

[32] H. Xu and B. Yu. Automatic thesaurus construction for spam filtering using revised back propagation neural network. *Expert Systems with Applications*, 37:18–23, 2010.

[33] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the International Conference on Machine Learning*, pages 412–420, 1997.

[34] K. Ying, S. Lin, Z. Lee, and Y. Lin. An ensemble approach applied to classify spam e-mails. *Expert Systems with Applications*, 37:2197–2201, 2010.

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

[36] B. Zhou, Y. Yao, and J. Luo. Cost-sensitive three-way email spam filtering. *Journal of Intelligent Information Systems*, 42:19–45, 2014.

[37] V. Zorkadis, D. A. Karras, and M. Panayotou. Efficient information theoretic strategies for classifier combination, feature extraction and performance evaluation in improving false positives and false negatives for spam e-mail filtering. *Neural Networks*, 18:799–807, 2005.