Using Huffman Trees in Features Selection to Enhance Performance in Spam Detection

Cleber K. Olivo[1], Altair O. Santin[1], Luiz E. S. Oliveira[2]

[1] Graduate Program in Computer Science – Pontifical Catholic University of Paraná (PUCPR)
[2] Department of Informatics – Federal University of Paraná (UFPR)
Curitiba – PR - Brazil

{cleber,santin}@ppgia.pucpr.br, lesoliveira@inf.ufpr.

Abstract. *Spam detection is very costly when compared to the simple task of spreading spam. Most approaches aim to reach higher accuracy percentages, leaving the classification performance in background, what may cause many problems, such as bottlenecks in the e-mail system, huge infrastructure investments and waste of resources pooling. To avoid these problems, this paper proposes a hierarchical spam features organization using Huffman Trees, where the most important features stay closer to the root. With the reduction of these trees (leaves pruning) the feature space is significantly reduced, speeding up the e-mail classification process. The experiments showed a performance 60 times faster when compared to Spam Assassin.*

1. Introduction

Spam is the term assigned to messages sent indiscriminately, usually without the consent of the recipient. Although this term was first attributed to a message sent to multiple users in an electronic medium [1], spam is not limited to its virtual form. However, because it can be easily sent, it is in the electronic media that spam causes the biggest problems. In recent decades, even with the popularization of other communication services, e-mail has prevailed as one of the most popular tools for user communication, making it the preferred service for spreading spam.

Historically, spam statistics have always been worrying. In 1997, AOL estimated that 5 to 30 percent of its 10 million e-mails received per day were spam [2]. In 2004, a study predicted that this percentage would reach 95 percent on a global scale by 2015 [3]. Statistics released by Symantec showed a percentage of 89.1 percent in 2010 [4], being the highest percentage until then [5]. The spam percentages have declined in recent years, with 66% in 2013, 60% in 2014 and 53% in 2015. However, it is estimated that the daily global volume of e-mails on the Internet daily is approximately 190 billion, with a forecast increase of 4% (more than 197 million) by the end of 2016 [6].

Spamming goes far beyond the user annoyance. Internet service providers and their users have a high cost, caused by bandwidth waste and the costs of the technologies used to detect it [1]. In some more serious cases, such as phishing, the damage done can go far beyond a simple annoyance, causing the users to compromise their computers or even having financial losses [7].

Most of the proposed approaches in the literature do not consider the performance (resource consumption) of the antispam system as a key factor of the e-mail system, focusing only on the classifier's accuracy. Usually a large number of features is used to achieve a good percentage of correctness in the classification. The greater the number of features, the greater the complexity of the classification model and the consumption of resources. When a large volume of e-mail is received in a short period, if the classification system is too complex, there may be a queuing of messages that can be perceived by users.

This paper proposes a way to reduce the complexity of the classification model without great impact on the classifier's accuracy. The frequency of the words is used as a feature in the classification model. Words are arranged hierarchically in a Huffman Tree, so that the most frequent words in spam messages are closer to the root of the tree, and the less frequent words closest to the leaves. The size of the tree can be reduced, reducing the complexity of the classification model. The presented results showed that this approach can be promising.

The paper is organized as follows. Section 2 presents an overview of the e-mail system, the spam detection as a textual problem and some of the main approaches proposed to mitigate the problem. Section 3 explains the Huffman tree generation, and how this structure can be used to detect spam. Section 4 details the implementation of the proposal and the obtained results. Section 5 shows related works. Finally, Section 6 presents the conclusions about the proposal.

2. The E-mail Spam

Spam detection techniques based on message content (e.g. text and images) usually involve the area of pattern recognition. The classification of messages using these techniques is usually done in the e-mail servers, before arriving in the user's mailbox.

2.1. The E-mail Service

Simple Mail Transfer Protocol (SMTP) is the standard protocol for e-mail transfer [8]. The rules for exchanging messages between e-mail servers are defined in RFC 2821, which specifies that e-mail consists of an envelope and a content [8]. The content is divided into two parts: header and body. The e-mail body can be composed of images and text using hypertext features such as HTML (Hypertext Markup Language). The body of the e-mail is structured according to the MIME format – Multipurpose Internet Mail Extensions [9].

The SMTP protocol itself has limitations that are exploited by spammers, making it possible, for example, to replace the address in the sender's field with a different one from the e-mail of whom is sending the message. However, it's in the e-mail body that spammers use most of their tricks to circumvent antispam mechanisms.

2.2. Spam as a Textual Classification Problem

Spammers often develop new techniques for spreading spam to circumvent detection techniques. In the message content, they involve technical subterfuges to circumvent the textual classification filters. These techniques range from the purposeful insertion of words, which confuse the e-mail classifier, to the use of technical subterfuges, such as hypertext language (HTML) features incorporated into the e-mail.

In order to confuse antispam tools based on the words frequency, spammers can intentionally insert some words that are also common for non-spam messages. This can occur, for example, with Bayesian classifiers [10], which rely on the most frequent words for both classes (spam or non-spam) to determine whether a message is spam.

A technique known as textual obfuscation performs the exchange exclusion, or insertion of characters in the words. This creates a new string of characters, but remains providing the same human understanding. The single word 'viagra' can have more than one sextillion of variations [11]. This is possible due to the large number of characters existing in the Unicode pattern, which offers 120,672 codes representing multi-lingual characters, ideograms and symbol collections (v8.0) [12]. The negative effect of textual obfuscation was demonstrated by Liu and Stamm [13], through an experiment where the substitution of characters totally impaired the classification of spam messages.

As it can be seen, there are several factors that make it difficult to classify e-mails, requiring classifiers to use refined techniques or even preprocessing steps before the classification. The refinement of these techniques has as an aggravating factor the consequent increase of computational cost.

2.3. Approaches for Textual Spam Detection

Pattern recognition consists of the automatic discovery of patterns in the data, using computational algorithms that allow to classify the data in different categories [14]. Some of the most used techniques used in the fight against spam are the Bayesian classifiers, Neural Networks, and Support Vector Machines.

Among the proposals that use word frequency for the classification of e-mails, Bayesian filters are among the most common ones. Bayesian decision theory is a fundamental statistical approach to the classification problem [10], with the Naïve Bayes (NB) classifier being one of the most used for textual classification [15]. Its simplicity makes it easy to be implemented, usually achieving good accuracy, when compared to other more sophisticated machine learning algorithms [16]. NB has been one of the most applied algorithms in commercial antispam tools [17]. Chen, C. and colleagues [17] compared three proposals using NB [18, 19, 20] and four other traditional approaches (SVM, C4.5, NB and KNN). They conclude that it is better an improved NB algorithm than traditional methods of classification (including NB itself).

Drucker, H., Wu, S., and Vapnik, V. N [21] presented a piece of work that, showed several of the main concepts used in detection techniques based on terms frequency, such as TF (Term Frequency) – number of times a word appears in a document, TF-IDF (Term Frequency - Inverse Document Frequency) – which defines the relevance of a term within a collection of documents, and Stop List – a list of terms that should not appear in the features vector. In addition, they also present several widely-used techniques to evaluate the performance of the classifier and validate the results, as well as to study the use of the SVM classifier in comparison with other classification algorithms (Ripper, Rocchio and Boosting Decision Trees).

Due to the variety of existing techniques, there are several papers that analyze or test several of these techniques in the e-mail classification [22, 23]. Although highly promising, the most commonly used classifiers become known to spammers, who may exploit certain limitations in important stages of classification. A spammer may send messages that intentionally contain frequent words in e-mails that are not spam, so these words would proportionally generate false positives if those messages are included in the training database. This technique, known as evasion [24], is often used in other security-related applications such as IDS (Intrusion Detection System) [25].

3. Hierarchical Features Organization

Most e-mail classification approaches are not concerned with making better use of the features used in the classification model. A reduction in the number of features can bring significant performance gains for the antispam system in terms of reducing the computational cost, avoiding problems such as message queuing or resource wastage. A way to reduce the features number is organizing them according to their occurrence.

Some of the most widely used and accepted ways to measure the occurrence of terms in textual documents are TF (Term Frequency), IDF (Inverse Document Frequency) and TF-IDF (multiplication of TF by IDF). These metrics measure the importance of certain words in a document [26]. Due to its simplicity and good results during preliminary tests, TF was chosen to conduct the experiments.

This section shows how a Huffman Tree can better organize the classifier's features space. Huffman encoding is a lossless data compression technique, i.e., when data decompression occurs, the result is a bit-by-bit sequence identical to the original data sequence [27]. The data organization is done through a binary tree, in which a distribution of the data occurs based on the occurrence of symbols. The most frequent symbols are positioned closer to the root of the tree. In the case of spam, the symbols used to distribute the data in a Huffman tree may be the words found in the messages. The distribution of these words is arranged in the tree, so that the most frequent words of the database are closer to the root and the less frequent words are closer to the leaves.

Figure. 1 represents the hierarchical distribution of the spam words, based on the frequency at which they occur, in a Huffman Tree. At the first step (i) the symbols (words) are given a weight, which represents the frequency at which they occur on the spam database. Words are ordered by their respective weights in an increasing way, and then each pair of words have their weights added together. The sums are always made with the leftmost pairs, grouping from the lowest weight to the largest one. The result is five small trees, with their roots having the sum of the weights of their leaves. After grouping, roots should be rearranged based on weight, if necessary.

In the next step (ii), a new grouping is done, but the root of greater number (weight 68) ends up being left over. Again, the trees are rearranged so that the weighted root 68 is moved to the middle (iii). A new sum of the weights of the roots is made, and this time the remaining one is the root with weight 108. Finally, the two remaining roots are rearranged and have their weights added, forming a single tree (Huffman Tree).

As words are arranged in the tree, it is possible to observe that the most frequent ones (result, please, new, viagra, look and best) are positioned at a level closer to the root than the less frequent words (report, success, price, offer).

This hierarchical distribution facilitates the feature selection process, where less frequent spam words are usually further from the root. These features are usually non-discriminatory and therefore do not have much use in the message classification process. A reduced number of features (a reduced tree) results in a simplified classification model and, therefore, less computationally costly. With multilevel distribution, the tree can be reduced (pruned) level by level, significantly reducing the number of features at each pruning, until the result of classification starts to be impaired. The remaining tree contains the most discriminating and most useful features (words) for the classifier.

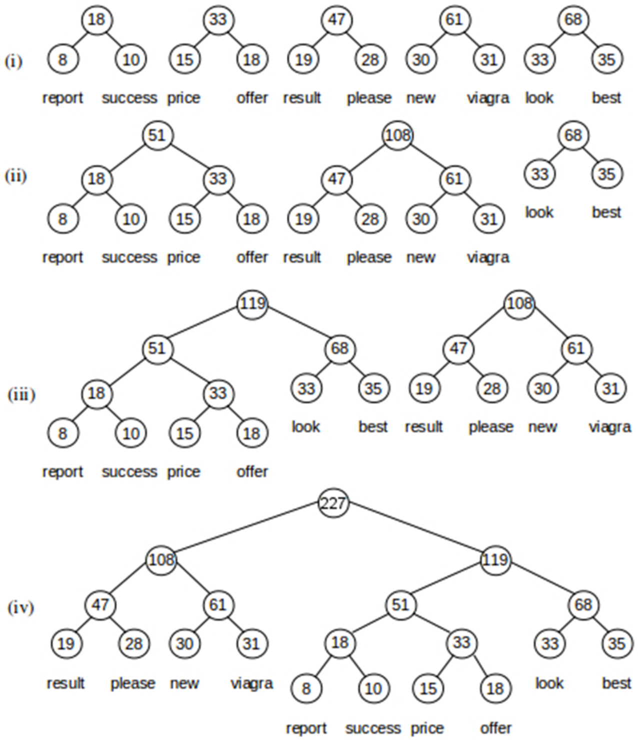

Figure 1. Feature Distribution in a Huffman Tree

4. Implementation of the Proposal

The proposal of this research was implemented in 4 main stages. The first step (section 4.1) involved the preparation of the databases and features extraction. The second step (section 4.2) is the e-mail classification, that also involves a gradual pruning of the Huffman Tree. In third step (section 4.3), performance tests were conducted. Finally, (section 4.4) the results of this approach were compared to Spam Assassin.

4.1. Database Preparation and Features Extraction

This proposal used different e-mail databases that are available online. One of the chosen databases was the Enron-Spam corpus [28, 29]. The Enron-Spam database consists of spam messages received by six users, with spam from various sources [30]. There is a total of 19,088 legitimate messages (not spam) and 32,988 spams. It is a public database very explored in the literature and still has non-spam messages. However, this is a very old database considering the evolution of spam over the years.

Thus, this proposal also used a second and a third database that were composed of e-mails from the database Untroubled Spam [31]. It is also available online and contains messages collected since 1998, but does not have non-spam e-mails. In addition, e-mails from users of mailing lists were collected on the Internet to compose the non-spam database. The second database (Untroubled 2015) consisted of 182,933 e-mails, of which 111,788 were spam (with spam of the whole year of 2015) and 71,145 were not spam e-mails. The third database (Untroubled 2016) consisted of 264,647 e-mails, with 212,216 spam messages (with spam of the whole year of 2016) and 52,431 non-spam messages. These numbers of spam messages were achieved after removing e-mails composed essentially of Asian charsets (e.g. KOI8-R, SHIFT-JIS, EUC-KR etc.), that are more than

700 thousand messages in the original 2015 and 2016 Untroubled databases. We consider that classifiers specialized in specific idioms are better than generalized classifiers. This proposal is focused in English language.

Prior to feature extraction, all the terms found in the training databases were validated after a search in an English dictionary. Terms that were not present in the dictionary were not considered as a feature. Additionally, all databases were preprocessed for e-mail header removal, stop-word removal, and suffix reduction.

The Python NLTK (Python Natural Language Toolkit) [32] was used for the removal of non-English words, stop-words and word stemming. Stop-words are terms that are among the most common in a language, for example articles and prepositions. The removal of these words is recommended before processing natural language, since they do not have much use in the classification process, and may even spoil the result. The reduction of suffixes (stemming) was done using the Snowball algorithm. For example, the words "receive" and "receiving", after being processed by the Snowball algorithm, both result in "receiv". Once the stop words removal and the word stemming were done, the features vectors were extracted using TF as a measurement of the words frequencies.

By containing all the "known" features of a spam database, the Huffman Tree was essential for the execution of the work. Therefore, it was of total importance that the Huffman tree was composed only by spam features collected in the training database. The features of non-spam messages were not used in this work. In other words, if a word appears only in e-mails that are not spam or spam messages in the test database, it does not appear in the tree of that classification model.

The Huffman Tree can also be used as a tool to see the more significant changes in spam features (most used terms) among bases or over the years. Table 1 lists the most common terms from the top 50 that are common to all the three bases. That is, after collecting the top 50 words for each base, there are 22 terms that common to all of them.

Considering the Enron Spam database is more than one third smaller than the Untroubled databases, for the training and test steps, similar to k-fold cross-validation, the database were divided into 10 parts (10-fold cross validation). In 10 steps, each part k is separated to test the model, and the remaining parts k-1 are used to train the model. Then a Huffman Tree was generated for each k-1, ensuring that there was a unique set of features and a distinct frequency of words for each step of the cross-validation. The experiments with Untroubled 2015 and 2016 databases were conducted with half of the messages for training the models and the other half for testing. The Untroubled Huffman trees were generated only with the training sets.

Table 1. Most Common terms from top 50 (occurring in all spam databases)

Also	Best	Day	Get	Go	Like	Look	Make
New	One	Order	See	Stop	Take	Time	Us
Use		Visit	Want	Way	Work	Year	

The Enron-Spam tree was composed of 13,159 words that were distributed in 12 different levels. In the trees formed with messages from both Untroubled Spam databases and mailing lists there were 9,067 and 10,161 features distributed in 13 different levels.

4.2. E-mail Classification

The classifier applied for the messages classification was Liblinear [33, 34]. When the number of instances and features is very large, the Liblinear classifier usually consumes less memory, being much faster and with a precision very similar to LibSVM [35], which is usually one of the best classifiers for the detection of spam [17, 21, 23]. Before creating the model, the best value to calibrate each classifier parameter was obtained. The classification method used was the primal-based support vector classification. Table 2 shows the accuracies after each Huffman Tree pruning. In a simple definition, accuracy represents the total of samples correctly classified.

Table 2. Accuracy rate after Huffman tree pruning

No. of Pruning	E-Mail Database		
	Enron	Untroubled 2015	Untroubled 2016
0	98.12	97.90	98.66
1	98.04	97.88	98.64
2	98.14	97.86	98.62
3	97.58	97.70	98.59
4	95.64	97.63	98.50
5	95.26	97.48	98.36
6	94.70	97.32	98.31
7	93.45	97.23	97.55
8	90.46	95.97	97.27
9	84.93	95.53	96.55
10	70.16	92.26	94.67
11	-	87.19	91.68
12	-	75.99	84.76

Note that, for the Enron database there was a decrease of only 0.54% after 3 consecutive pruning steps, which proves that the essential features remain at the top of the Huffman Tree. After 4 consecutive prunings the accuracy drops more than 1%. With 11 prunings the result is inconclusive due to the lack of features for classification.

For the 2015 database, the result after the third pruning decreases only 0.2%. The accuracy kept above 97% even after the seventh pruning. The Untroubled 2016 had a result with only 0.16% lower than the original tree after 4 prunings.

Table 3 shows the number of terms present in the tree after each pruning step. One can be noticed that, for example, in the Enron database, with a tree divided in 9 levels (after three prunings), the accuracy of the classifier drops only 0.54%, but with a reduction of more than 71% in the number of features). Reducing the number of features results in a lighter classifier, providing significant performance gains, as will be shown in Section 4.3. Moreover, an effective feature selection leads to parsimonious classifiers that require less memory and are faster to train and test, and can also reduce the feature extraction costs and lead to better generalization [36].

Table 3. Quantity of Features After Tree Reductions

Tree Size	E-mail Database		
	Enron	Untroubled 2015	Untroubled 2016
13	-	9066	10160
12	13159	8544	9219
11	11566	6867	7420
10	7424	5487	5459
9	3744	4206	4118
8	2402	3111	3147
7	1578	2430	2423
6	1021	1900	1940
5	633	1500	1487
4	338	967	794
3	124	333	349
2	27	80	102
1	1	10	15

If the system administrator considers an accuracy of 97% as acceptable, the reduction of the feature space is even greater for the Untroubled bases, reaching a reduction of 79% (a tree of 6 levels for 2015) and 85% (a tree of 5 levels for 2016).

Table 4. False Positives and False Negatives

Prunings	Enron		Untroubled 2015		Untroubled 2016	
	FP Rate	FN Rate	FP Rate	FN Rate	FP Rate	FN Rate
0	0,013	0,022	0,036	0,012	0,041	0,007
1	0,012	0,024	0,036	0,012	0,042	0,007
2	0,012	0,022	0,036	0,012	0,041	0,007
3	0,010	0,032	0,037	0,014	0,042	0,007
4	0,056	0,036	0,038	0,014	0,044	0,008
5	0,060	0,040	0,039	0,016	0,046	0,009
6	0,068	0,044	0,040	0,018	0,047	0,009
7	0,071	0,062	0,041	0,019	0,063	0,015
8	0,129	0,076	0,053	0,032	0,066	0,018
9	0,175	0,137	0,056	0,038	0,074	0,025
10	0,492	0,186	0,083	0,074	0,096	0,043
11	-	-	0,151	0,113	0,165	0,063
12	-	-	0,329	0,128	0,333	0,087

As a complementary evaluation, the false positives (FP) and false negatives (FN) are also important to choose the classifier. The values, of these rates are shown in table 4. Note that for the Enron Spam database the FP and FN rates remain practically unchanged even after the occurrence of two prunings. If the administrator considers the false positives a bigger problem than false negatives, the FP rate drops slightly after three prunings. However, it will result in a worst FN rate. Untroubled bases had minimal variations after five or six prunings.

The Enron Spam test database was formed with 3299 spam messages and 1909 non-spam messages. The Untroubled 2015 test database was formed with 55,895 spams and 35,573 non-spam and the Untroubled 2016 test database was formed with 106,111 spam and 26,216 non-spam.

4.3. Performance Tests

The performance tests showed significant performance gains after the reduction of the number of features used by the classifiers, as can be seen in table 5. The values were obtained computing an arithmetic mean, after repeating the experiments 10 times. The coefficient of variation was below 3%. Since the time required for training and classification of e-mails depends heavily on the environment (software and hardware) in which it occurs and the number of classified messages, the time required for the original classifiers (without reduction of features) is considered as 100%. The execution times of the classifiers that suffered reduction in the number of features are presented as a fraction of the time of the original classifier, also in percentages.

The real times took to complete the testing steps (with the original feature space – represented as 100% in table 5) had an average of 65ms, 1s171ms and 1s205ms, respectively. These times refer only to the classification process (after preprocessing and feature extraction). All the performance tests were performed in the same environment and with the same conditions: Intel® CoreTM i7 4510U (4 threads, 2 GHz up to 3.1GHz, 4MB L3 Cache), 8GB RAM and the operating system GNU/Linux Mint v17.3.

At the Enron database, it was observed that with three prunings the percentage of correctness remained almost the same, without significant changes in terms of false positives and false negatives, and the reduction in the number of features was more than 71%. Additionally, it takes 77% of the time of the original tree, as can be seen in table 5. This time reduction of 23% in the test stage (most crucial step in a real environment) could avoid a bottleneck in the e-mail system and also the waste of resources.

The Untroubled 2015 had a reduction of only 2.65% of time in the testing stage after the third pruning and Untroubled 2016 had a reduction of 5.58% (with minimal FP x FN changes). However, supposing the e-mail server is receiving an amount of e-mails bigger than it can handle, the classifier model could be switched to a simpler one that obeys a minimal requirement, i.e., an accuracy above 97%. This requirement allows to apply the feature sets with seven and eight prunings, representing a time reduction of almost 13% and 21%. After the contingency, the antispam system could make use of the more complex classifier again.

It is important to emphasize that, although the increase in performance is one of the main contributions of this research, the accuracy of the classifier is not in the background. Obviously, the main effort was to demonstrate the effect of non-discriminating features on the cost of the classification process (there was no great effort to raise the accuracy of the classifier). For example, accepting an accuracy of 98% does not mean ignoring 2% of e-mails. The lighter classifier could be used only at a critical time, in which the server would have many messages queued, until the situation is normalized. However, opting for a faster classifier with almost the same results would be the most correct and proactive decision to make.

Table 5. Performance results (testing step)

Pruning	Enron	Untroubled 2015	Untroubled 2016
0	100,00	100,00	100,00
1	95,41	99,95	98,33
2	90,31	98,96	97,90
3	77,04	97,35	96,29
4	73,47	95,22	94,42
5	68,88	93,13	92,16
6	64,29	90,69	89,74
7	58,16	87,11	85,25
8	46,94	80,11	79,11
9	30,61	67,60	63,11
10	17,86	40,29	44,55
11	11,22	18,27	24,21
12	-	7,42	9,87

4.4. Tests with Spam Assassin

Tests with Spam Assassin [37] were conducted with two different purposes: i) compare the e-mail classification accuracy of this approach with a well-known antispam tool; ii) find some criteria to check if this approach has an acceptable performance at training and classification stages, even without any code improvement.

Spam Assassin reached an accuracy of 99.90% on the Enron database, 99.86% on Untroubled 2015 and 99.56% on Untroubled 2016. The tool threshold was changed (manually) to obtain the best possible result. These results evidence the feasibility of the proposed approach, once a better accuracy could be reached testing another classifier or using another feature representation (e.g. TF-IDF).

The performance comparison was done by sampling, using a total of 2 thousand e-mails, which were half spam and half non-spam. This step took 10s294ms to perform both classes training. The testing stage, where classification occurs, was executed with the reading of each e-mail of the testing dataset and took 471s434ms to complete.

The training stage of this approach involves the header and stop-words removal, dictionary checking, word stemming, TF calculation, creation of the Huffman Tree, feature extraction and creation of the classification model by LIBLINEAR. All these tasks took 10s728ms. At the testing stage, the e-mails of the test dataset were preprocessed likewise the training stage and the feature vectors were composed. The features vectors were submitted to the classifier, finishing the testing process in 7s743ms. When comparing the real time took for this testing step (with preprocessing and feature extraction) to the real time took by the classification job (section 4.3), it is evident that the heaviest work is previous to the classification itself.

As an overview of the results, the training process of this approach was 4.21% times slower than SpamAssassin, but the testing stage was 60 times faster. It must be considered that SpamAssassin is a very robust antispam tool and the time calculation does not consider its internal processes during training and classification. However, it should also be considered that this approach was coded in Python language, without any special programming treatment to improve performance. Moreover, the main difference in time occurs at the testing stage, which is the more important (computational costly) step in real environments to prevent bottlenecks in e-mail ingoing. Another advantage of this approach is that these times can be reduced even more with the tree pruning, as shown in table 5.

5. Related Works

The feature subset selection is a procedure which can reduce not only the cost of pattern recognition, but provides a better classification accuracy in some cases [39]. In spam detection as a problem of textual classification, the feature space has a high dimension. A feature space with a large number of terms can be unsuitable for a classifier or even decrease the categorization accuracy, once the ambiguity of meaning of terms can also prohibit the classifier from choosing the categories deterministically [40].

Meng, J. et. al. (2011) proposed a feature selection method named feature contribution degree (FCD). The FCD calculation considers that the features have different distributions in different categories. When a feature has a larger value of FCD, the feature probably has a stronger distinction capacity for categorization. Thus, if the feature has a larger value of FCD, it has a better chance of being selected [40].

Wiratunga et al. evaluated the use of generalization after feature selection in a spam base (LingSpam corpus). The results of generalization combined with boosting outperformed the other algorithms, with the highest accuracy approaching 99% with only 110 features. However, the authors suggest the domain is relatively easy because the LingSpam corpus has a few very discriminatory features for non-spam messages that are sufficient to differentiate spam messages [41].

Trivedi and Dey tested the Genetic feature search and Greedy Stepwise feature search techniques. For the Enron dataset, with only 48 best features out of 1500 initially created features, the Genetic Search reached an accuracy of 87.1% and the Greedy Search reached an accuracy of 94.2% [42].

Most of the related work propose improvements in the feature selection method. There is not much information about the impacts of feature selection in e-mail classification performance, being the main efforts concentrated in classifiers accuracy. Furthermore, when studying spam, a critical issue is the possibility to observe how it changes over the time (due to the spammer tricks or seasonality), what is possible using a tool such as the Huffman Tree.

6. Conclusions

This approach has as main objective the reduction in the time necessary for the spam detection, without significant lost in the accuracy of the classification. This avoids wasting resource pooling, required infrastructure investments, and is also useful in situations such as message queuing caused by the antispam system. Although the accuracy of the classifier is not the main contribution, the results obtained were very close to what was achieved with Spam Assassin. Thus, better results would probably be obtained if there was a special effort to do it.

The features organization in a Huffman Tree is a simple and efficient solution and can be easily implemented by hierarchically organizing the features of spam with the most relevant terms quickly accessed (i.e., positioned closer to the root). This form of organization could also, through the observation of trees obtained in different months or years, analyze in a different way the changes in the main words of spam in different epochs. The use of trees of different sizes to train the classifier is an approach that demonstrates a lot of feasibility and flexibility if used in a real environment.

The proposed method of using Huffman Trees to organize and reduce the feature space could also be easily implemented for another idiom such as Portuguese. The Huffman Tree could be reconstructed or updated as necessary, for example, if the administrator perceives an increase of false positives or false negatives.

In general, the main objectives of this work were achieved. It was evidenced that a waste occurs in the number of features used in the classification models. Most of these features are non-discriminatory and, therefore, have no use for the classifier. The performance tests found the wastage in processing time when no type of feature selection is performed. Complementary tests also have shown that the classification phase is 60 times faster than the well-known Spam Assassin tool.

7. References

[1] K. Kleiner. *Happy Spamyversary! Spam Reaches 30* [Online]. Available: https://www.newscientist.com/article/dn13777-happy-spamiversary-spam-reaches-30/

[2] B. Hoanca, "How good are our weapons in the spam wars?", IEEE Technology and Society Magazine, vol. 25, issue 1, pp. 22-30, 2006.

[3] B. Whitworth and E. Whitworth, "Spam and the social technical gap", IEEE Computer, vol. 37, issue 10, pp. 38-45, 2004.

[4] Symantec, “January 2011 Intelligence Report” [Online]. Available: https://www.navixia.com/images/pdf/newsletter/MLI_2011_01_January_Final_en-us.pdf

[5] Symantec, “May 2013 Intelligence Report” [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/security-center/archives/intelligence-report-may-13-en.pdf

[6] Symantec, “Internet Security Threat Report” [Online]. Rep. 21, Apr. 2016. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

[7] Olivo, C. K.; Santin, A. O.; Oliveira, L. S., “Obtaining the Threat Model for E-mail Phishing”, Applied Soft Computing, vol. 13, issue 12, pp. 4841-4848, 2013.

[8] J. Klensin. (2001, April). *RFC 2821 - Simple Mail Transfer Protocol* [Online]. Available: http://www.ietf.org/rfc/rfc2821.txt

[9] N. Freed and I. Borenstein. (1996, November). *RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* [Online]. Available: http://tools.ietf.org/rfc/rfc2045.txt

[10] R. Duda, P. Hart and D. Stork, *Pattern Classification*, 2nd edition, Wiley-Interscience, 2000.

[11] *There are 600,426,974,379,824,381,952 ways to spell Viagra* [Online], Available: http://cockeyed.com/lessons/viagra/viagra.html

[12] *The Unicode Standard – Technical Introduction* [Online]. Available: http://www.unicode.org/standard/principles.html

[13] C. Liu and S. Stamm, "Fighting Unicode-Obfuscated Spam", Proceedings of the Anti-Phishing Working Group - 2nd Annual eCrime Researchers Summit, pp. 45-59, ACM, 2007.

[14] C. Bishop, *Pattern Recognition and Machine Learning*, 1st edition, Springer, 2007.

[15] K. Schneider, "A comparison of event models for Naive Bayes anti-spam e-mail filtering", Proceedings of the 10th conference on European chapter of the Association for Computational Linguistics, vol. 1, pp. 307-314, ACM, 2003.

[16] I. Androutsopoulos, G. Paliouras and E. Michelakis, "Learning to Filter Unsolicited Commercial E-Mail" [Online], NCSR "Demokritos" Technical Rep. 2004/2, Mar. 2004. Available: http://nlp.cs.aueb.gr/pubs/TR2004_updated.pdf

[17] C. Chen, Y. Tian and C. Zhang, "Spam Filtering with Several Novel Bayesian Classifiers", IEEE 19th International Conference on Pattern Recognition, pp. 1-4, 2008.

[18] E. Frank, M. Hall and B. Pfahringer, "Locally Weighted Naive Bayes", Proceedings of the 19th conference on Uncertainty in Artificial Intelligence, ACM, pp. 249-256, 2002.

[19] H. Zhang, L. Jiang and J. Su, "Hidden Naive Bayes", Proceedings of the 20th National Conference on Artificial Intelligence, vol. 2, ACM, pp. 919-914, 2005.

[20] G. Webb, J. Boughton, and Z. Wang, "Not so Naive Bayes: Aggregating One-Dependence Estimators", Machine Learning, vol. 58, issue 1, pp. 5-24, 2005.

[21] H. Drucker, S. Wu and V. Vapnik, "Support Vector Machines for Spam Categorization", IEEE Transactions on Neural Networks, vol. 10, issue 5, pp. 1048-1054, 1999.

[22] R. S. S. Kiran and I. Atmosukarto, "Spam or Not Spam – That is the Question" [Online], Technical Report, University of Washington, 2005. Available: http://courses.cs.washington.edu/courses/cse573/04au/Project/mini2/Ravi&Indri/spamfilter_ravi_indri.pdf

[23] T. Guzella and W. Caminhas, "A Review of Machine Learning Approaches to Spam Filtering", Expert Systems with Applications, Elsevier, vol. 36, issue 7, pp. 10206-10222, 2009.

[24] B. Nelson, B. Rubinstein, L. Huang, A. Joseph and J. Tygar, "Classifier Evasion: Models and Open Problems", Privacy and Security Issues in Data Mining and Machine Learning, vol. 6549, Lecture Notes in Computer Science, p. 92-98, Springer, 2011.

[25] M. Barreno, B. Nelson, R. Sears, A. Joseph and J. Tygar, "Can Machine Learning be Secure?", Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, pp. 16-25, 2006.

[26] *"TF-IDF: A Single Page Tutorial – Information Retrieval and Text Mining* [Online]. Available: http://www.tfidf.com

[27] M. Sharma, “Compression Using Huffman Coding”, International Journal of Computer Science and Network Security, vol. 10 no.5, pp. 133-141, 2010.

[28] V. Metsis, I. Androutsopoulos and G. Paliouras, "Spam Filtering with Naive Bayes – Which Naive Bayes?", Proceedings of the 3rd Conference on E-mail and Anti-Spam (CEAS 2006), Mountain View, CA, USA, 2006.

[29] *The Enron-Spam Datasets* [Online], Available: http://www.aueb.gr/users/ion/data/enron-spam

[30] G. Cormack, “E-mail Spam Filtering: A Systematic Review”, Foundations and Trends in Information Retrieval, vol. 1, n.4, pp. 335-455, 2006.

[31] *Untroubled Spam Archive* [Online]. Available: http://untroubled.org/spam/

[32] *Natural Language Toolkit – NLTK 3.0 documentation* [Online]. Available: http://www.nltk.org

[33] R. Fan, K. Chang, C. Hsieh, X. Wang and C. Lin, “LIBLINEAR: A Library for Large Linear Classification”, Journal of Machine Learning Research, vol. 9, 2008.

[34] *LIBLINEAR: A Library for Large Linear Classification* [Online]. Available: https://www.csie.ntu.edu.tw/~cjlin/liblinear

[35] C. Hsu, C. Chang and C. Lin. *A Practical Guide to Support Vector Classification* [Online]. Available: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[36] Z. Xu, G. Huang and K. Weinberger, "Gradient Boosted Feature Selection", Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 522-531, 2014.

[37] *SpamAssassin – The #1 Enterprise Open-Source Spam Filter* [Online], Available: http://spamassassin.apache.org

[38] *SpamAssassin Configuration File* [Online]. Available: http://spamassassin.apache.org/full/3.3.x/doc/Mail_SpamAssassin_Conf.html#language_options

[39] A. Jain and D. Zongker, "Feature Selection: Evaluation, Application, and Small Sample Performance", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, issue 2, pp. 153-158, 1997.

[40] J. Meng, H. Lin and Y. Yu, "A Two-stage Feature Selection Method for Text Categorization", Computers and Mathematics with Applications, Elsevier, pp. 2793-2800, 2011.

[41] N. Wiratunga, I. Koychev and S. Massie, "Feature Selection and Generalization for Retrieval of Textual Cases", Proceedings of the 7th European Conference on Case-Based Reasoning, Springer Verlag, pp. 806-820.

[42] S. Trivedi and S. Dey, "Effect of Feature Selection Methods on Machine Learning Classifiers for Detecting Email Spams", Proceedings of the 2013 Research in Adaptive and Convergent Systems, ACM, pp. 35-40, 2013.