

# SPAM FILTERING USING COMPRESSION

M. Farmer<sup>1</sup>, G. Richard<sup>1,2</sup>

*British Institute for Technology and E-commerce 1  
Avicenna House 258-262 Romford Road London UK  
{farmer,grichard}@bite.ac.uk \**

F. Faure<sup>2</sup>, M. Lopusniac<sup>2</sup>

*Institut de Recherche en Informatique de Toulouse 2  
118 rte de Narbonne 31062 Toulouse France  
richard@irit.fr*

## ABSTRACT

One of the most irrelevant side effects of the e-commerce technology is the development of spamming as an e-marketing technique. Spam emails (or unsolicited commercial emails) induce a burden for everybody having an electronic mailbox: detecting and filtering spam is then a challenging task and a lot of approaches have been developed to identify a spam before it is posted in the end user mailbox. In this paper, we focus on a relatively new approach whose foundations rely on the works of A. Kolmogorov. The main idea is to give a formal meaning to the notion of “information content” and to provide a measure of this content. Using such a quantitative approach, it becomes possible to define a distance which is a major tool for classification purpose. To validate our approach, we proceed in two steps: first we use the classical compression distance over a mix of spam and legitimate emails to check out if they can be properly clustered without any supervision. It has been the case, highlighting a kind of underlying structure for the spam emails. In a second step, we implement a k-nearest neighbors algorithm providing 85% as accuracy rate. Coupled with other anti-spam techniques, compression-based methods could bring a great help in the spam filtering challenge.

## KEYWORDS

Spam, Kolmogorov complexity, compression, clustering, k-nearest neighbors

## 1. INTRODUCTION

Information security is one of the main conditions to insure a safe electronic marketplace. This is achieved through the use of multiple techniques to prevent unauthorized use. Encryption, authentication/password protection, and login policies all provide some level of security. It is well known that, when accessing certain web sites, we become obvious target for spammers. A spam is just a (commercial) unsolicited email and can be considered as dangerous as soon as we reply to it. If we do not reply, it is only a kind of rubbish mail polluting your mailbox and we need to spend some time to select between relevant emails and spams. Simple calculations show that, with the increasing number of spam received every day, the cost for business in term of time and money is not negligible: simply because when you are filtering your mailbox, you do not perform the task you are supposed to do. That is why there is a real need for automatic filters. But detecting and filtering spam is a quite difficult challenge due to the dynamic nature of junk e-mail. A truly effective spam filter must block the maximum unwanted e-mail, with the minimum number of false positives (messages, wrongly identified as spam). Of course, this is further complicated by the fact individual users have different views on what they consider to be a spam: a spam is definitely a personal subject. The most basic methods are quite similar (see [Graham2002] for a survey) in term of philosophy: checking content for textual analysis, black lists (which list has to be regularly updated) of hosts and domains, content analysis, etc. For instance, one compares the sender domain with a black list of domain names known as issuing spam emails. This simple version can be improved by adding an analysis of the body of the email and looking for specific keywords (dictionary approach). The user can define his

personal dictionary and tune the system for his personal use. From a network infrastructure viewpoint, the system classifying the incoming emails can be:

- on the server side (spamassassin, MailScanner). Emails are then tagged when they arrive on the server without taking into account individual user profile.
- on the client side and in that case, the filtering system is generally included as a plug-in within the mail client (Outlook express, Thunderbird, Eudora, etc...).

Whatever the chosen architecture, the problem of filtering the spam emails still remains a classification problem and methods issued from the machine learning field are quite effective. From this viewpoint, the main philosophy is to train a learner on a sample set of mails (the witness set) clearly identified as junk or not junk and then to use the output of the system as a classifier for the next incoming emails. One of the most sophisticated approaches to deal with spam email is the so-called Bayesian technique. This approach is based on the probabilistic notion of Bayesian network (see [Pearl2003] for an in-depth definition) and is considered as one of the most efficient technique, providing few of false positive. It seems that the future of anti-spamming systems relies on a mix between traditional practices (black lists, dictionaries, etc...) and techniques issued from the field of machine learning.

When it comes to classification, the mathematical concept of distance is always very helpful. From a heuristic viewpoint, if 2 objects are close from each other with regard to a suitable distance, they should be similar in some sense. Roughly speaking, if a mail  $m$  is close to a junk mail, then  $m$  is likely a junk mail as well. That is why there are a lot of attempts to define a distance suitable to a particular situation. In our case, these distances generally take into account parameters such as the domain of the sender, the occurrence of specific words, etc... An email is then represented as a point in a vector space and using a proper distance and the previous heuristic, we can identify spam emails. Despite the fact that the origin of an email can obviously give some hints about its status, we basically think that a spam email is defined by its content: then, in our approach, we decide to focus on the body of an email. An email is classified as spam because the informative content of its body is the informative content (or close to) of a spam email. So we are interested in a distance between 2 emails insuring that, if they are close, we know that they have a similar informative content. It remains to give a formal meaning to the concept of "informative content". This is not a new problem and Kolmogorov (or Kolmogorov-Chaitin) (see [Kolmogorov1965]) complexity is a strong candidate for this purpose. The main idea behind Kolmogorov works is to consider the informative content of a string  $s$  as the measure of the size of the ultimate compression of  $s$ ,  $K(s)$ . Surprisingly, complexity plays a critical role in security and can be broadly applied as the basis for security analysis and design. For instance, in the field of network security [Kulkarni2001,Bush2002], the advantage of using complexity as a fundamental parameter to monitor system health and achieve information safety lies in its objectivity. Any given string (or file) has a Kolmogorov complexity without regard to the details of the system on which it is running. The operating system, protocol being used, and meaning or type of the data represented by a particular string, while related to string complexity, need not be known in order to measure complexity. Tools based upon this paradigm do not require detailed a priori information about known attacks, but rather compute vulnerability based upon an inherent, underlying property of information itself, namely its Kolmogorov complexity. In this paper, we investigate the use of Kolmogorov complexity as a tool to classify junk mails without any analysis of the content and without any analysis of the header. The remaining of this document is structured as follows: in section 2, we give a flavor of the Kolmogorov theory and its very meaning. In section 3, we describe the main idea behind practical applications of Kolmogorov which is the definition of a suitable information distance. Since  $K(s)$  is an ideal number, section 4 explains how to estimate  $K$  and the relationship with compression techniques. In sections 5, 6 and 7, we first experiment the compression distance to cluster a mixture of spam and legitimate emails: this shows that there is a common underlying structure in spam emails, then we describe our implementation, our experiments and results. We discuss our approach as well. Finally, we conclude in section 8.

## 2. KOLMOGOROV COMPLEXITY

Starting from the fact that all the information we deal with a computer are digitalized, it is relevant to consider every object as a finite string of 0 and 1. Kolmogorov complexity  $K(x)$  is a measure of the

descriptive complexity contained in an object or string  $x$ . It refers to the minimum length of a program such that a universal computer can generate a specific string  $x$ . A good introduction to Kolmogorov Complexity can be found in [Bennett1998] with a solid treatment in [Ming1997]. Kolmogorov complexity differs from Shannon entropy in that it relates to the specific string being considered rather than the source distribution. Kolmogorov complexity can be roughly described as follows, where  $T$  represents a universal computer (Turing machine),  $p$  represents a program, and  $x$  represents a string:

**$K(x)$  is the size  $|p|$  of the smallest program  $p$  such that  $T(p) = x$**

We can thus consider  $p$  as the essence of  $x$ : there is no shorter program than  $p$  to output  $x$  and we can view  $p$  as the most compressed form of  $x$ . The size of  $p$ ,  $K(x)$ , can thus be considered as the ultimate amount of information contained in  $x$ . Having said that,  $K(x)$  becomes the lower bound limit of all the possible compressions of  $x$ : Kolmogorov complexity of a string  $x$  should be considered as the ultimate compression size of the string  $x$ . Since every data can be coded as a binary string, we can say that every data has a Kolmogorov complexity which is the complexity of its digital representation. Random strings have rather high Kolmogorov complexity on the order of their length, as patterns cannot be discerned to reduce the size of a program generating such a string. On the other hand, strings with a large amount of structure have fairly low complexity. Universal computers can be equated through programs of constant length, thus a mapping can be made between universal computers of different types: the Kolmogorov complexity of a given string on two computers differs by known or determinable constants. In some sense, this complexity is a kind of universal characteristic attached to a given data. For a given data  $x$ ,  $K(x)$  is a strange number which cannot be computed just because it is an ideal lower limit related to an ideal machine (Universal Turing machine or calculator). And this is a major difficulty which can be overcome since the length of any program producing  $x$  is an upper bound of  $K(x)$ . We will develop this fact in section 4. But now, we have to understand how we can build a suitable distance starting from  $K$ : this is the aim of the next section.

### 3. INFORMATION DISTANCE

As previously explained, from a data mining or knowledge discovery viewpoint, mathematical distances are powerful tools to classify new data. In our case, given an incoming email  $s$ , we want to compute the distance between  $s$  and another email  $s'$ , previously classified (as junk or not junk): if this distance is small enough, we could decide that  $s$  belongs to the same class and transmit this information to the end user.

Starting from Kolmogorov complexity, it is relatively easy to define a distance called "Information Distance" or "Bennett's distance" (see [Bennett1998] for a complete study). The informal idea is as follows: given  $a$  and  $b$ , two strings or files, we can put  $K(a) = K(a-b) + K(a \cap b)$  and  $K(b) = K(b-a) + K(a \cap b)$ .

The first equation simply says that the complexity of  $a$  (the information content) is the sum of the proper information content of  $a$ , denoted  $a-b$  and the common content with  $b$  denoted  $a \cap b$ . Now, we concatenate  $a$  with  $b$  and we get a new file denoted  $a.b$ . If we compute  $K(a.b)$ , we get  $K(a-b) + K(b-a) + K(a \cap b)$  since there is no redundancy with Kolmogorov compression. So the number  $m(a,b) = K(a) + K(b) - K(a.b)$  is just a measure of the common information content to  $a$  and  $b$ : this is a very approximate reasoning leading to two very formal notions:

$$d(a,b) = m(a,b) - \min(K(a), K(b))$$

and

$$d_N(a,b) = (m(a,b) - \min(K(a), K(b))) / \max(K(a), K(b)) = 1 - m(a,b) / \max(K(a), K(b))$$

Both of them are distances but  $d_N$  is normalized and takes into account the relative size of  $a$  and  $b$ . Let us understand the very meaning of  $d_N$ . If  $a=b$ , then  $K(a) = K(b)$  and  $m(a,b) = K(a)$  thus  $d_N(a, b) = 0$ . On the opposite side, if  $a$  and  $b$  do not have any common information,  $a \cap b$  is empty and  $K(a.b) = K(a) + K(b)$  thus  $m(a,b) = 0$  and then  $d_N(a, b) = 1$ . Formally speaking,  $d_N$  is a (semi) distance over the set of finite strings:  $d$  is called the normalized information distance or the Bennett distance. If  $d_N(a,b)$  is very small, it means  $a$  and  $b$  are very similar, if  $d_N(a,b)$  is close to 1, it means  $a$  and  $b$  have very few information in common. Let us examine in the next section how we can use  $K$  in the real life!

## 4. COMPLEXITY ESTIMATION

Now that we have a formal tool, it remains to see how it can be practically used for real business purpose. As discussed above, exact measurement of Kolmogorov complexity is not achievable, but if we remember that  $\mathbf{K}(\mathbf{x})$  is the lower limit of all the compressions of  $\mathbf{x}$ , we understand that every compression  $\mathbf{C}(\mathbf{x})$  of  $\mathbf{x}$  gives an estimation of  $\mathbf{K}(\mathbf{x})$ . Back to the theoretical framework, a decompression algorithm is considered as our universal Turing machine  $\mathbf{T}$  such that  $\mathbf{T}(\mathbf{C}(\mathbf{x})) = \mathbf{x}$ . It is now clear that we are looking for lossless compression algorithms and fortunately, we have a variety of such algorithms coming from the “compression” community. It is out of the scope of this paper to investigate that field and we just give a quick overview of some classical compression tools available on the market. On one side, we have the formal algorithms: Lempel-Ziv-Welch, Huffman, Burrows-Wheeler. All of them are lossless compression techniques which is exactly what we need. On the other side, we have the real implementations adding some clever manipulations to the previous algorithms:

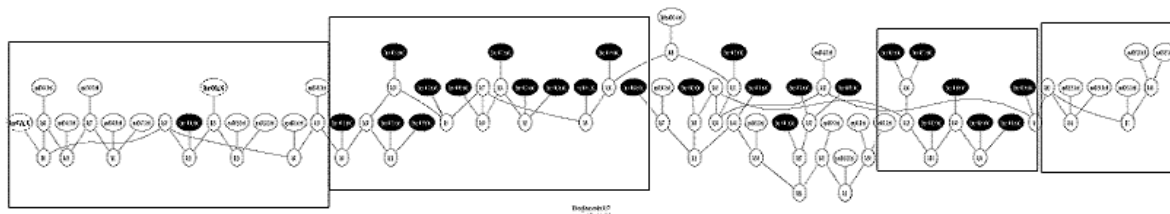
- . Unix `compress` utility based on LZ
- . `zip` and `gzip` which are a combination of LZ and Huffman encoding (for instance, Adobe Reader contains an implementation of LZW)
- . `bzip2` which first uses Burrows-Wheeler transform then a Huffman coding.

In term of compression ratio, generally we have the following inequality (where LZMA(x) is a variant of LZ):  $0 < \mathbf{K}(\mathbf{x}) < |\text{LZMA}(\mathbf{x})| < |\text{bzip2}(\mathbf{x})| < |\text{gzip}(\mathbf{x})|$ . Instead of dealing with the ideal number  $\mathbf{K}(\mathbf{x})$ , we will deal with the size of the compressed file. The better the algorithm compresses the data  $\mathbf{x}$ , the better the estimation of  $\mathbf{K}(\mathbf{x})$ . When replacing  $\mathbf{K}(\mathbf{x})$  by  $|\mathbf{C}(\mathbf{x})|$  where  $\mathbf{C}$  is one of the previous compressor, it becomes quite clear that the previous information distance is not any more a true mathematical distance but remains sufficient for our purpose. Today, LZ-based algorithms are considered among the most efficient ones for text compression and as a consequence, they are the suitable tools for our implementation.

## 5. COMPRESSION FOR CLUSTERING

Before to go on with our idea, we need to have at least a flavor of the power of the compression distance to discriminate between legitimate and spam emails, without any supervision. To do that, we use the technique of [Cilibrasi2005] that we briefly recall here in the context of spam filtering:

- we start from a mix of 50 spam and legitimate mails.
- for each couple of mails  $(\mathbf{m}, \mathbf{m}')$ , we compute the distance  $\mathbf{d}(\mathbf{m}, \mathbf{m}')$ : we get a symmetric matrix.
- we provide a 2 dimensional graphic representation of the 50 emails as a binary tree respecting the relative distance between the emails. Since this is not possible in a 2 dimensional view, the generated tree  $T$  is associated to a confidence level  $S(T)$  between 0 and 1, describing how well the tree represents the original data. There is no real difficulty here since we use the works and tools of [Cilibrasi2005] to handle our data. It is well known that clustering using quartet tree heuristic is not tractable. It took around 20 hours (on a standard laptop) to get the tree  $T$  below where our test set contains 50 emails and  $S(T) = 0.917$  which is quite acceptable. To provide a 2 dimensional view, we used Graphviz freely available on <http://www.graphviz.org/> (mainly developed within AT&T lab.). The tree we got is below where we have darkened the spam emails.



We can clearly distinguish 4 homogeneous areas (delimited with rectangles) although the last one (without delimitation) is quite mixed. Of course, it would be interesting to do the same experience with a bigger test set but the execution time would be prohibitive. Anyway, we consider this picture, highlighting a basic unsupervised clustering of our test set, as sufficiently indicative to implement a classification process using compression distance technique.

## 6. OUR IMPLEMENTATION

In order to validate our approach, we start from a classical k-nearest-neighbours (k-nn) algorithm where all neighbours equally contribute to the determination of the value of the class variable. Using [D'Amato2004] extension, we weight the contribution of each of the k-nn according to their distance to the new case, giving greater weight to closer neighbours. The witness set S is constituted with both spam and normal emails. Let us describe a little bit more our definition: given a mail  $m_i$  in the witness set, we compute  $d(m, m_i)$  and if  $d(m, m_i) \neq 0$ , we define  $w(m_i) = 1/d(m, m_i)$  (this is the weight of the mail  $m_i$ ).

Now we have to distinguish between 2 cases:

1. There are some neighbours  $m_i$  such that  $d(m, m_i)=0$  and then

$$P(\mathbf{m} \text{ is a spam}) = (1/k) * (\# \{ m_i \mid d(m, m_i)=0 \text{ and } m_i \text{ is a spam} \})$$

$$\text{Of course } P(\mathbf{m} \text{ is legitimate}) = (1/k) * (\# \{ m_i \mid d(m, m_i)=0 \text{ and } m_i \text{ is legitimate} \}) = 1 - P(\mathbf{m} \text{ is a spam}).$$

2. All the k nearest neighbours  $m_i$  of  $m$  are such that  $d(m, m_i) \neq 0$  and then we compute the probability for  $m$  to be a spam as:

$$P(\mathbf{m} \text{ is a spam}) = A/B \text{ where}$$

$$A=(1/k) * (\sum w(m_i) \mid m_i \text{ is a spam}) * \# \{ m_i \mid m_i \text{ is a spam} \} \text{ and}$$

$$B = A + A' \text{ where } A' = (1/k) * (\sum w(m_i) \mid m_i \text{ is legitimate}) * \# \{ m_i \mid m_i \text{ is legitimate} \}$$

It is clear that  $P(\mathbf{m} \text{ is legitimate})=A'/B$ . Normalization by the summation  $B$  of all weighted class probabilities is required to guarantee that the above measure  $P$  satisfies the properties of a probability measure. By weighting distances, we insure that very distant examples will have little effect on the classification of the incoming email  $m$ . When  $m$  has some null distance neighbours, then we consider only these neighbours because they are viewed as the best representatives of the class of  $m$ . As a complexity estimator, we use LZW algorithm whose C (go through <http://www.complearn.org> for a complete package). This technique performs without information loss and is quite good in term of runtime efficiency. Our implementation gives the choice between the weighted version of k-nn and the classical one despite the fact that the weighted one is generally more accurate. Basically, starting from an incoming email, we compute its k-nn belonging to a set of witness emails, previously classified as junk/not junk. Then, we choose the class (according to the previous weighted formula) as the class for the incoming email. The distance we use is just the information distance estimated through LZW compression. Our algorithm is implemented in C and our programs (source and executable) are freely available on request.

## 7. RESULTS AND DISCUSSION

### 7.1 First experience with the non normalized distance d

Using the previous software, our experimentation has been done with a set of 150 incoming emails to classify. We then performed 4 series of tests using 4 different witness sets as below:

**25junk/25legitimate ----50junk/50legitimate----75junk/75legitimate----100junk/100legitimate**

The only parameter to be tuned is k and we choose k as 11, 21, 31, 41, 51, 61, 71, 81 and 91 when we have enough witness mails. That is why with only 50 witness mails, we go up to 49 only. We provide only the weighted version results since the other version of k-nn (without weight) gives lower accuracy. Below we provide 4 graphics which summarize our results (false positive means a legitimate email classified as spam):

Table 1: 25junk/25legitimate witness mails

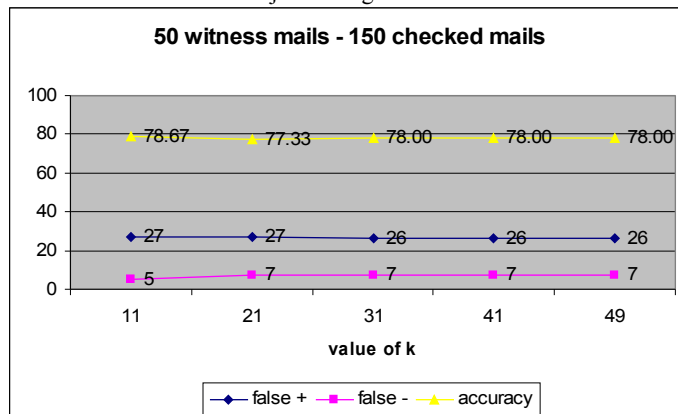


Table 2: 50junk/50legitimate witness mails

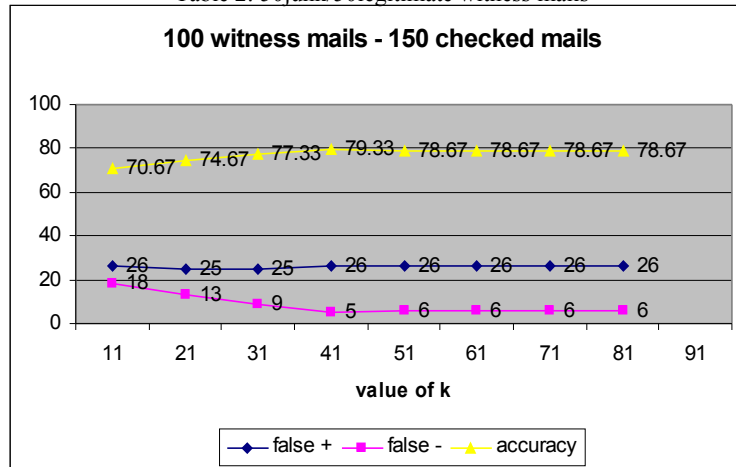


Table 3: 75junk/75legitimate witness mails

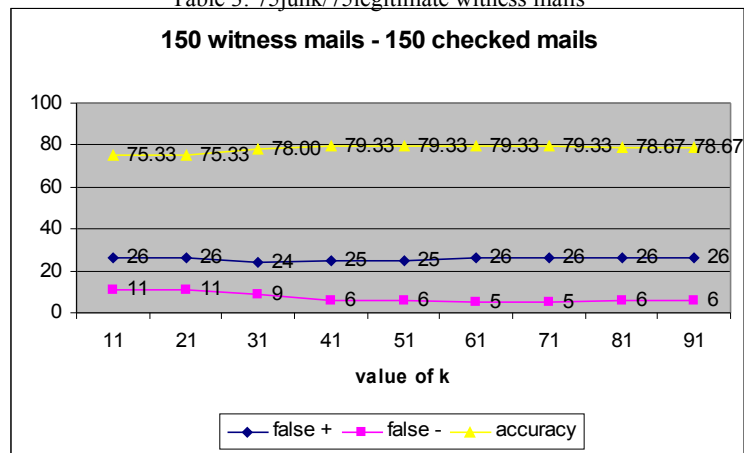
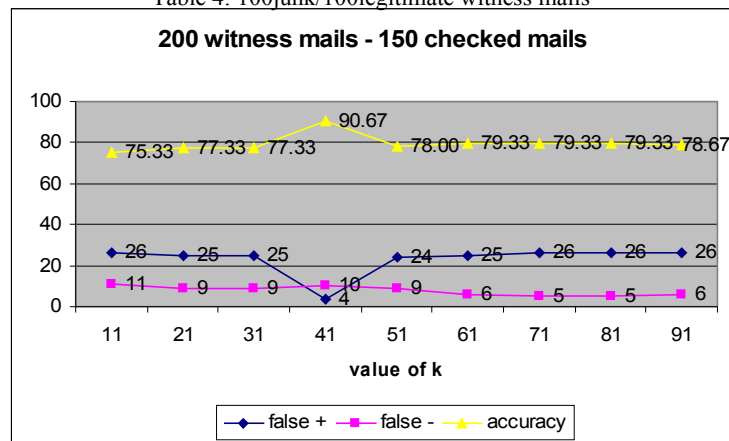


Table 4: 100junk/100legitimate witness mails



Visual inspection of our curves provides clear evidence that information distance really captures an underlying “semantics”. Since in our samples, we have mails in diverse languages (French, English), we can think that there is a kind of common underlying information structure for a spam. Except in one case (k=41 and 200 witness mails), the accuracy rate remains in the range of 80%, whatever the number of witness mails, and the level of false negative remains low. So your mailbox remains clean. As usual with anti-

spamming techniques, the level of false positives means that you have to examine your junk folder from time to time. It is interesting to note that we get roughly the same results with only 100 witnesses: in that case, there is no need to have too many witnesses in our database to get better results. Let us note that the runtime is about 14mn to classify 150 emails (on a standard PC) when we have 200 witness mails. Of course, this kind of algorithm, considered as plugged into a mail client (Thunderbird, Outlook, etc...) will have to classify on-the-fly incoming email and in that case, this is quite a fast process.

## 7.2 Second experience with the normalized distance $d_N$

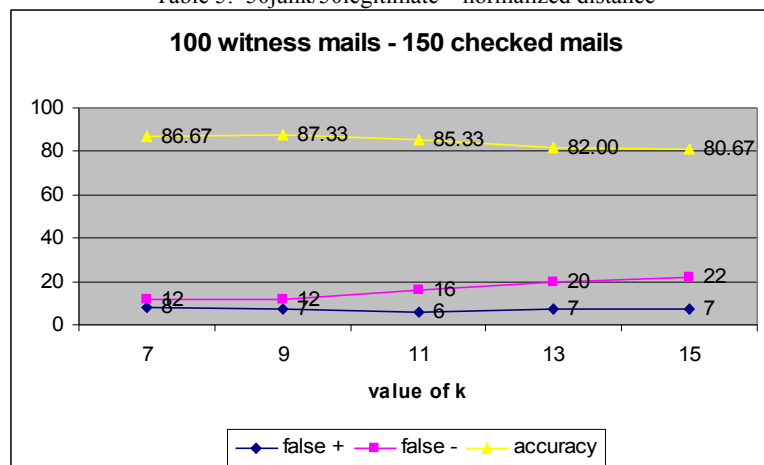
Based on our previous batch, we decided to work on a fixed set of 100 witnesses and 150 incoming emails to filter. Our sets (both witness and test) are different from the previous experience. We have optimized our process in diverse ways:

i) From a theoretical viewpoint,  $k$  can vary between 1 and the cardinal of the witness set. An empirical limit is just the square root of the witness set: in our case exactly 10. So we decide to experiment with  $k = 7, 9, 11, 13$  and 15 which are numbers in the range of 10.

ii) We modified the compression distance to take into account the relative size of the mails: instead of using the compression distance  $d$ , we use the normalized compression distance  $d_N$ .

Below are graphical views of our results (weighted and non weighted version provide the same results):

Table 5: 50junk/50legitimate – normalized distance



It is quite clear that the normalized distance coupled with a clever choice of  $k$ , performs better than the standard one. With regard to the previous experiences, the most interesting features are:

- The accuracy rate is now around 86% instead of 80% previously.
- The number of false positives is now below the number of false negatives which is a sought-after feature for a spam filtering system.
- The best accuracy rate is obtained with  $k=9$  and decreases when  $k$  increases: there is no need to look for more neighbours than the square root of the witness set.

It is interesting to note that the weighted version of  $k$ -nn does not perform better than the non weighted one. At the moment, it is not clear if this is due to our particular sample set or if this is a general property when using the normalized distance. We have to investigate in order to find out a kind of optimal number of witness mails. There is probably no universal theoretical value for such a parameter but, with more experiments, we should be able to provide evidence for a realistic empirical value.

## 8. CONCLUSION AND FUTURE WORKS

Despite their relative novelty, the complexity-based approaches have been proved quite successful in numerous fields of IT: data mining, classification, clustering, fraud detection, etc. Within the field of IT security, we can refer to the works of [Kulkarni2001], [Bush2002], [Bush2003] and more recently [Wehner2006]: these works are mainly devoted to analyze data flow coming from network activities and to

detect intrusion or virus. In this paper, we investigate the field of anti-spamming and we consider Kolmogorov complexity  $K(\mathbf{m})$  of an email  $\mathbf{m}$  as a tool to classify it as *junk* or *legitimate*. Instead of dealing with a sophisticated clustering algorithm, we have implemented a weighted version of k-nearest neighbours, using the information distance deduced from  $K$ . Despite the fact that our tests have been limited, the first results we have obtained clearly show that the compression distance is meaningful in that situation. This is not completely coming as a surprise: the seminal works of [Bennett1998] then [Cilibrasi2005] have showed the way and have highlighted the real power of compression methods. In our context, one of its main advantages is that there is no need to deeply analyze the diverse parts of the mail (header and body). We can of course refine our algorithm but simply having a more powerful compression technique would probably bring more accurate results. We have to notice that current spam starts to come in gifs and pdfs (already compressed): in that case, our method is irrelevant and has to be adapted.

Another advantage of this kind of “blind” techniques is that there is no need to update a dictionary or a black list of domain name. The system automatically updates when the witness base evolves over time (i.e. the database of junk/legitimate mails which is basically unique to a given user): it can be trained on a per-user basis, like Bayesian spam filtering. Another similarity with Bayesian filtering is that complexity based methods do not bring any explanations about their results. Fortunately, this is generally not a requirement for a mail filtering system: in that particular case, it cannot be considered as a drawback. This is definitely a point that should be investigated more deeply in the future. Finally, in order to quickly validate the complexity approach for spam filtering, we have chosen to use a k-nn algorithm as a classifier. Nevertheless, it is well known that more sophisticated approaches could be used, for instance the support vector machine technique which is known to be very successful for classification purpose. It is quite clear that these kinds of approach are not the ultimate Graal but we strongly believed that, when mixed with other classical strategies, they will definitely bring a step further in the field of spam filtering. For us, it remains to investigate how to combine the complexity with the other information (header, domain, etc...) to provide a more accurate system and to ultimately tend through a “spam-free” internet!

## REFERENCES

- Bennett C. et al., 1998. Information distance, *IEEE Transaction on Information Theory*, 44:4, pp. 1407-1423.
- Bush S., and Stephen F., 2002. Active Virtual Network Management Prediction: Complexity as a Framework for Prediction, Optimization, and Assurance, *Proceedings of the 2002 DARPA Active Networks Conference and Exposition*, IEEE Computer Society Press, San Francisco, California, USA, pp. 534-553.
- Bush S. and Stephen F., 2003. Extended Abstract:Complexity and Vulnerability Analysis, *Complexity and Inference*, 2-5, , DIMACS Center, Rutgers University.
- Cilibrasi R. and Vitányi P., 2005. Clustering by compression. *IEEE Trans. Information Theory*, 51:4.
- D’Amato C. et al., 2004. Extending the knn classification algorithm to symbolic objects, <http://citeseer.ist.psu.edu> and *Proceeding of ECML/PKDD*, Italy.
- Graham P., 2002. A plan for spam, <http://www.paulgraham.com/spam.html>.
- Kolmogorov A. N., 1965. 3 approaches to the quantitative definition of information, *Prob. Inform.Transm*, 1:1, pp. 1-7.
- Kulkarni P. and Bush. S., 2001. Active network management and kolmogorov complexity, *OpenArch*, Anchorage Alaska.
- Li M.and Vitányi P., 1997. *Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag,. ISBN 0-387-94053-7.
- Pearl J. and Russell S., 2003. *Bayesian Networks*, in M. A. Arbib (Ed.), *Handbook of Brain Theory and Neural Networks*, pp. 157–160, Cambridge, MA: MIT Press.
- Wehner S., 2006. Analyzing worms and network traffic using compression, <http://arxiv.org/abs/cs.CV/0504045>.