

A Complexity-based Method for Anti-Spamming

F. Faure², M. Lopusniac², G. Richard^{1,2}, M. Farmer¹

¹ *British Institute of Technology and E-commerce,
87-89 Plashet Road, London E13 0RA, UK*

² *Institut de Recherche en Informatique de Toulouse,
118 route de Narbonne, Toulouse, 31000, France*

*{farmer, grichard}@bite.ac.uk
richard@irit.fr*

Abstract

A huge diversity of approaches is in use to deal with network security and spam in particular. 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 “information content” and to use it to measure in some sense the deviation with regard to standard behaviour. We apply this kind of method to detect spam email, which is a well-known disease for mail client. To validate our approach, we have implemented a k -nearest neighbors algorithm which could easily become a plug-in for an open source mail client. Despite its strong theoretical basis, Kolmogorov’s works give rise to very simple practical applications, easy to implement and with surprisingly accurate results.

1. Introduction

Currently, information security is achieved through the use of multiple techniques to prevent unauthorized use. Encryption, authentication, password protection, and policies all provide some level of security against unauthorized use. Detecting and filtering spam (i.e. commercial unsolicited emails) creates a number of complex challenges 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 spam. The basic methods are quite similar (see [2] for a survey): checking content for textual analysis, black lists of hosts and domains, content analysis, etc. For instance, a simple idea to classify an email is to compare the sender domain

with a black list of domain names known as issuing spam emails. The black list has to be regularly updated. This simple version can be improved by adding an analysis of the body of the email and looking for specific keywords based on a dictionary approach. The user can define his proper dictionary so tuning the system to his profile. Then, when a mail is classified as junk mail, it is the responsibility of the client to generate the suitable action. From a network infrastructure viewpoint, the system classifying the incoming emails can be:

on the server side (spamassassin for instance is set up with standard Linux distribution). Emails are 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...).

It is a good practice to set up an anti-spam on the main mail server and to use a mail client that incorporates a filtering system. Whatever the chosen architecture, the problem of eliminating the spam emails still remains a clustering problem and it is quite natural to turn to methods issued from the machine learning field. 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 [23] for an in-depth definition) and is considered as one of the most powerful till today (such a system is implemented with Thunderbird). It seems that the future of anti-spamming systems relies on a mix between traditional practice (black lists, dictionaries,

etc...) and techniques issued from the field of machine learning.

When it comes to classification, there is a mathematical concept which is always very helpful: the concept of distance. From an intuitive viewpoint, if 2 objects are close from each other, 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. Generally these distances 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, we can identify spam emails. Despite the fact that the origin (the envelop) of an email can give some hints about its status, we basically think that a spam email is defined by its content. 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". Kolmogorov (or Kolmogorov-Chaitin) 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 , $K(s)$.

In this paper, our aim is to demonstrate that Kolmogorov complexity and its associated distance can be used as a tool to classified junk mails:

- without any analysis of the content
- without any analysis of the header

The remaining of this document is structured as follows: in section 2, we give a brief review 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 is an ideal number, section 4 explains how to estimate K and the relationship with compression techniques. In the two following sections 5 and 6, we describe our implementation and results. We discuss our approach as well. Finally, we conclude in section 7.

2. What is Kolmogorov complexity

Starting from the fact that all the information we deal with in 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 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 is contained in [19] with a solid treatment in [13]. Kolmogorov complexity is

related to Shannon entropy, in that the expected value of $K(x)$ for a random sequence is approximately the entropy of the source distribution for the process generating the sequence. However, Kolmogorov complexity differs from 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 since there is no way to reduce p to get 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 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 should be considered as the ultimate compression size of the string. 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, and 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 by using the fact that 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 decide or to classify a 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 sufficiently small (i.e. less than a given threshold), 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 [17] for a complete study). The informal idea is as follows: given a and b, two strings or files, we can say:

$$\begin{aligned} \mathbf{K(a)} &= \mathbf{K(a-b)} + \mathbf{K(a \cap b)} \\ \mathbf{K(b)} &= \mathbf{K(b-a)} + \mathbf{K(a \cap b)} \end{aligned}$$

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 ∩ b.

Now, we can concatenate a with b and we get a new file denoted a.b. If we compute K(a.b) we get:

$$\mathbf{K(a-b)} + \mathbf{K(b-a)} + \mathbf{K(a \cap b)}$$

since there is no redundancy with Kolmogorov compression. So the number

$$\mathbf{m(a,b)} = \mathbf{K(a)} + \mathbf{K(b)} - \mathbf{K(a.b)}$$

is just a measure of the common information content to a and b. This is a very approximate reasoning leading to a very formal notion:

$$\mathbf{d(a,b)} = \mathbf{1 - m(a,b) / \max(K(a), K(b))}$$

Let us understand the very meaning of d. If a=b, then K(a) = K(b) and m(a,b) = K(a) thus d(a, b) = 0. On the opposite side, if a and b do not have any common information, a ∩ b is empty and K(a.b) = K(a) + K(b) thus m(a,b) = 0 and then d(a, b) = 1. Formally speaking, d is a (semi) distance over the set of finite strings: d is called the information distance or the Bennett distance. If d(a,b) is very small, it means a and b are very similar, if d(a,b) is close to 1, it means a and b have very few information in common. This is exactly what we need to start with classification. 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 K(x) is the lower limit of all the compressions of x, we understand that every compression C(x) of x gives an estimation of K(x). Back to the theoretical framework, a decompression algorithm is considered as our universal Turing machine T such that:

$$\mathbf{T(C(x)) = x}$$

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 [14], Huffman [15], Burrows-Wheeler [16]. 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):

$$\mathbf{0 < K(x) < |LZMA(x)| < |bzip2(x)| < |gzip(x)|}$$

Instead of dealing with the ideal number K(x), we will deal with the size of the compressed file. The better the algorithm compresses the data x, the better the estimation of K(x). When replacing K(x) by C(x), 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. 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 [21] extension, we weight the contribution of each of the k neighbours 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) ≠ 0, we define w(m_i) = 1/ d(m, m_i).

Now we distinguish between 2 cases:

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

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

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

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

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

and

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

Normalization by the summation B of all weighted class probabilities is required to guarantee that the above measure satisfies 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 . Our implementation gives the choice between the weighted version and the classical one despite the fact that the weighted one is generally more accurate. As a complexity estimator, we use LZW algorithm whose C source code is freely available (go through <http://www.complearn.org> for instance more complete package). This technique performs without information loss and is quite good in term of runtime efficiency.

Basically, starting from an incoming email, we compute its k -nearest neighbours 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.

6. Results and discussion

Using the previous software, our experimentation has been done with a set of 150 new 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 emails.

The only parameter we tune 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 the 4 graphics which summarize our results.

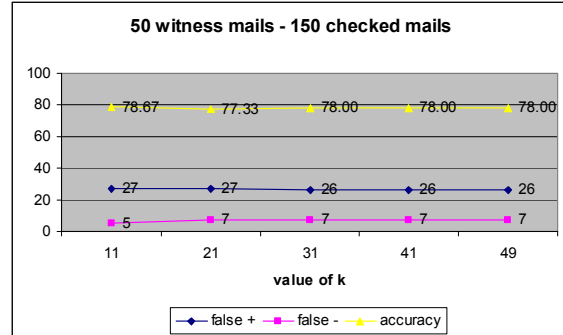


Table 1: 25junk/25legitimate witness mails

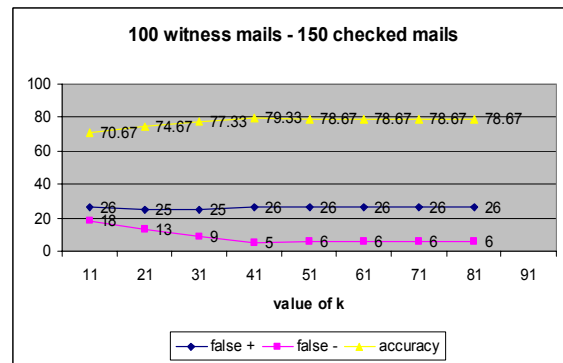


Table 2: 50junk/50legitimate witness mails

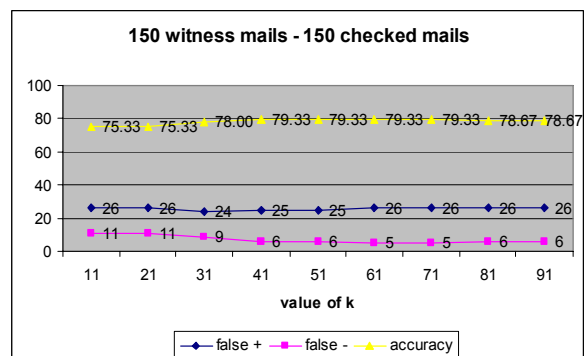


Table 3: 75junk/75legitimate witness mails

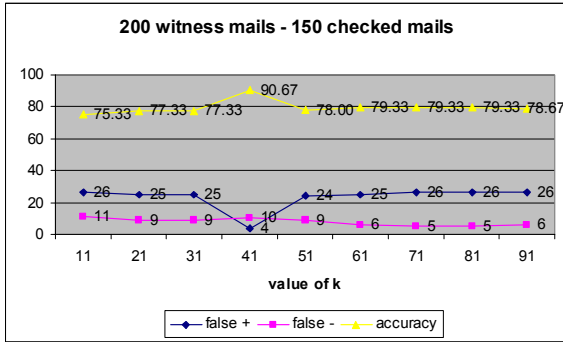


Table 4: 100junk/100legitimate witness mails

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 approximatively 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.. This runtime does not really rely on K since we just have a table of distances to check: the main influence is coming from the size of the witness set (here 200 emails): to compute the distance, we have to open and then to close 200 files exactly 150 times and this is a quite time-consuming operation. When checking 150 emails with a witness set of 50 spam and 50 legitimate mails, the runtime becomes more acceptable with around 6mn. 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.

We have now 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.

From a more general viewpoint, 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 [1,3], 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 of the data represented by a particular string, while related to string complexity, need not be known in order to measure string 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-Chaitin complexity. This is exactly the strategy we have implemented in our works.

7. Future works and Conclusion

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 [1], [3], [4] and more recently [20]: 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 $K(m)$ of an email 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 as distance 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 [17] then [18] have showed the way and have highlighted the real practical power of compression methods. 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.

Of course, multiple complexity estimators could be combined in order to improve discrimination accuracy.

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/not junk 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. On the dark side, it seems that the Bayesian methods perform better for the

false positive (they are almost “false positive” free). This is definitely a point that should be investigated more deeply in the future. Despite some commercial software ensure 99% of success in spam elimination (see <http://www.liveprism.com/> for instance), it is well known that the end users are not entirely satisfied by the current performances of anti-spam. In [23] for instance, people estimate around 15% the amount of daily spam they receive after the spam filters have done their job. It means 85% of accuracy which is a little bit better than what we get in our experiments. 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” email!

8. References

- [1] Kulkarni P. and Bush. S., *Active network management and kolmogorov complexity*, 2001. OpenArch 2001, Anchorage Alaska.
- [2] Graham P., *A plan for spam*, <http://www.paulgraham.com/spam.html>. August 2002
- [3] Bush, Stephen F., *Active Virtual Network Management Prediction: Complexity as a Framework for Prediction, Optimization, and Assurance*, Proceedings of the 2002 DARPA Active Networks Conference and Exposition (DANCE 2002), IEEE Computer Society Press, pp. 534-553, ISBN 0-7695-1564-9, May 29-30, 2002, San Francisco, California, USA.
- [4] Bush, Stephen F., Extended Abstract: *Complexity and Vulnerability Analysis, Complexity and Inference*, June 2-5, 2003, DIMACS Center, Rutgers University
- [12] Kirchner W., Li M., and Vitányi P., *The Miraculous Universal Distribution*. The Mathematical Intelligencer, Springer-Verlag, New York, Vol. 19, No. 4, 1997.
- [13] Ming Li and Paul Vitányi. *Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, 1997. ISBN 0-387-94053-7
- [14] Welch T., *A Technique for High Performance Data Compression*, IEEE Computer, June 1984 (v17.6)
- [15] Huffman D., *A method for the Construction of Minimum Redundancy Codes*, Proceeding of the IRE, September 1952
- [16] Burrows M and Wheeler D, *A block sorting lossless data compression algorithm*, Technical Report 124, Digital Equipment Corporation, 1994
- [17] Bennett C., Gacs P., Li Ming, Vitanyi P., *Information distance*, IEEE Transaction on Information Theory, 44:4,pp. 1407-1423, 1998
- [18] Cilibrasi R., Vitanyi P., *Clustering by Compression*, see <http://homepages.cwi.nl/~paulv>, 2004
- [19] Kolmogorov A. N., *Three approaches to the quantitative definition of information*, Problems Inform. Transmission, 1:1,pp. 1-7, 1965
- [20] Wehner S., *Analyzing worms and network traffic using compression*, <http://arxiv.org/abs/cs.CV/0504045>, 2006
- [21] D’Amato C., Esposito F. and al., *Extending the knn classification algorithm to symbolic objects*, <http://citeseer.ist.psu.edu> and proceeding of ECML/PKDD, Italy, 2004
- [22] Pearl J., Russell S., *Bayesian Networks*, in M. A. Arbib (Ed.), *Handbook of Brain Theory and Neural Networks*, pp. 157–160, Cambridge, MA: MIT Press, 2003.
- [23] *Anti-Spam Products give unsatisfactory Performance* <http://www.spamfighter.com/News-8811-Anti-Spam-Products-Give-Unsatisfactory-P>