EMAIL SPAM DETECTION
FEAT THOMAS BAYES

Gaspard GOUPY

Tecnológico de Monterrey, Puebla, México

Abstract. Have you ever heard about spam?
Spam is a brand of canned cooked pork made by Hormel Foods Corporation. I hear you asking "Ok, but why are you talking about food in a machine learning paper?". Because spam is also the noun given to any kind of unwanted, unsolicited digital communication, often an email, that gets sent out in bulk. Spamming quickly became a major problem in network communications. It is time consuming, but could also be dangerous for the users. However, thanks to machine learning, it is now possible to build systems able to detect spam. Today, these algorithms are well implemented in the majority of applications, such as social networks, instant messaging or email services. Nevertheless, do you know how this system works exactly? This is what this research paper is dealing with.
The purpose of this report is to present a model able to classify emails, using a content-based approach, into two categories: spam or human. The project has been realized for the course Intelligent Systems, taught by the Dr. Octavio Loyola-González at Tecnologico de Monterrey, campus Puebla. It is linked to the Kaggle competition "Detecting Spam Emails" created for the course. Java language is used for the text analysis and file processing and the software Weka for the classification. The solution developed is a system using the Naïve Bayes Multinomial Text classifier of the Weka software, with a training email dataset from the Kaggle course competition. Many algorithms and methods have been tested with different features, and the paper will present the progress and results until the final solution.

Keywords: security · spam detection · machine learning · classification

1 INTRODUCTION

With the spread of the internet, a new kind of communication has appeared: electronic mail. Also known as e-mail, or email, an electronic mail is a message sent through communication networks [3]. It entered limited use in the 1960s, but it was only possible to send messages to users on the same computer. Later in 1971, Ray Tomlinson has been credited as the inventor of electronic mail. Indeed, he developed the first system able to send emails between users on different hosts across the ARPANET [5]. The electronic address has been defined as: username @ host-name . domain (Example: `user@aol.com`). Now, there are many email providers offering their services across the world. Among the most

famous, there are gmail, outlook, hotmail, yahoo or aol. All of them are proposing services such as receiving, sending, and storing electronic messages. Some providers are free, others are focusing on privacy and security (like protonmail), or also on simplicity (like gmail). But they are all giving the possibility to communicate almost instantly with other users on the internet.

However, this facility to send and receive messages has been rapidly deviated from its original purpose. Indeed, in the 1990s, commercial use of the internet became possible and marketers began to send emails with marketing goals. Spam has been created. Email spam, also referred to as junk email, is unsolicited messages sent in bulk mostly by email (spamming) [4]. In 2014, around 90% of the email traffic was spam [6]. Even if this percentage went down, it still represents approximately 60% of the global email traffic [1], where the number of global e-mail users amounted to 3.9 billion in 2019 [2].

In this paper, the suspicious behaviors of spam emails is studied in order to build a system able to detect spam. The goal is to apply machine learning to identify automatically if an email has been written by a human or if it is a spam. A training and a testing dataset from the Kaggle competition is used, containing respectively 4135 and 1034 emails. Results are obtained with the Kaggle classification score. Java language is utilized to process the files and to send results to Kaggle (using its API).

2 RELATED WORKS

The development of the internet and networks has been very useful to facilitate communications between people. However, this facility to communicate is also a complication for user's privacy and security. Indeed, spam are encountered all around the internet: on social networks, websites, emails but also through sms messages and calls. They could deal with marketing purpose, invading the user's privacy, coming in by dozens daily. But they could also be dangerous: phishing, scamming, or malware [8]. This is why spam detection has been studied for a long time. It is a complicated work, in the way that spammer methods are changing with the time, on the platform spread, etc.

One of the first kind of spam that has emerged was the web spam [7]. The aim was to fill a website with keywords without any correlation with the content just to appear on the top of web searches. TrustRank has been rapidly introduced, in 2004, by researchers of Stanford University and Yahoo in the paper Combating Web Spam with TrustRank [13]. The algorithm was analyzing a website and returning a score which was its trust degree. The number varied between 0 (spam) and 10. Later in 2005, Google company registered the trademark, and now this algorithm is used by most of the search engines.

For spam detection in other applications, Yu-Sung Wu presented an approach in 2009 to detect spam calls, named spam detection in Voice-over-IP calls [9]. This is based on the semi-supervised learning methods with an improved algorithm called MPCK-Means. Another application of spam detection could be find in the paper Detecting Spammers and Content Promoters in Online Video Social Networks [10]. The authors has been studying videos on YouTube about spammers and promoters. A supervised classification algorithm has been proposed to classify and detect a spam video. Finally, in 2010, Alex Hai Wang used a machine learning approach to study spam bots detection on Twitter in the paper: Don't Follow Me - Spam Detection in Twitter [12].

We have seen that spam detection has been present for a long time and in a lot of applications. Now, what about email spam detection? In 1998, some researchers were the first to apply a Bayesian approach to filter spam emails, in the paper: A Bayesian Approach to Filtering Junk E-Mail [11]. The results showed that the classifier obtained a good score only by considering email raw text content, but was still better when analysing in addition domain-specific features. Now, Bayesian algorithm is considered as one of the best method to detect spam, and it is implemented on a lot of modern email services. There are a many works about spam email detection on the internet. In fact, it is often a study case for IT students. Nonetheless, there are different approaches to classify mails. Among them, we can find:

- Content-based approaches
- Header-based approaches
- Protocol-based approaches
- Approaches based on sender authentication
- Approaches based on social networks
- Trust and reputation approaches

3 OUR PROPOSAL

The system proposed to detect spam emails is a classification only based the text content feature, using the Naïve Bayes Multinomial Text classifier, from Weka software. This algorithm is designed for text data, and just operates on string. Indeed, it is based on word appearance only, it can account for multiple repetitions of a word and treats common words differently from unusual ones. It is also faster than plain Naïve Bayes. The classifier is set with most of default settings. Many configurations has been tested and the most accurate one uses:

- wordTokenizer
- Rainbow
- minimum word frequency of 3
- no stemmer

WordTokenizer is a method that tokenize the strings. Rainbow is a stop words handler program. The classifier ignores any words that don't occur at least 3

times. Before classifying, a class balancer algorithm has been applied, because the training dataset was imbalanced: 2954 real emails for 1181 spam. It is also important to understand that the proposed solution is very different from all the other tests and has been applied at the end of the project.

The formula of the multinomial Naïve Bayes Multinomial text classifier is :

$$= N! \times \prod_{i=1}^{k} \frac{P_i^{n_i}}{n_i!} \tag{1}$$

- N = n1+n2+...+nk is number of words in the text
- Pi is the probability of word i over all the text
- Ni is number of times it appears in the text

4 EXPERIMENTAL SETUP

To build a system able to classify emails with content-based approach using machine learning, the first thing required is a training dataset, which contains emails already classified (spam or human). A testing dataset (containing emails not classified) is also used to classify emails and report results to Kaggle website. The both files were encountered on the Kaggle course competition. Then, features were added to the email rows in both files, in order to make a classification according to these values. MeaningCloud and ParallelDots APIs were used to process text contents to add sentiment and emotion features. Self made algorithms were also used for other attributes, for a total of 18 features, presented in the table 1. Note: # is number.

Table 1. List of features used for the project

Feature ID	Feature	Feature ID	Feature	Feature ID	Feature
1	Score Tag	2	Agreement	3	Irony
4	Confidence	5	Subjectivity	6	Fear
7	Excited	8	Bored	9	Happy
10	Angry	11	Sad	12	# of characters
13	# of symbols	14	Word Density	15	# of unique words
16	# of links	17	# of unique links	18	# of letter words

The weka software is utilized for the classification. This is an open-source machine learning software which provides many functions such as file preprocessing, classification, data visualization, etc. The last part is about testing the classification results. Kaggle competition provides a way to submit the testing file previously downloaded, but with the class column filled. It compares the results to the real ones and gives a score according to the number of instances well classified. In this way, Kaggle API is used to automatize the submission

process. Once the weka result is saved in a file, a java program is started: it connects email row ID to its weka result, parses the file and uploads it to the Kaggle competition.

5 EXPERIMENTAL RESULTS AND DISCUSSION

5.1 First classification tests

Before finding the best solution, many tests have been made with different methods and features. Kaggle counts 150 entries for the classification results. The first submission was 3 months before the writing of this paper. The table 2 regroups a part of the first tests realized. Notes: Features numbers are corresponding to the IDs in the table 1. [x-y] means all the features included between the IDs x and y.

Table 2. First classification tests

Classifier	Features used	Kaggle result
K-NN	[1-10]	0.56155
J48	[1-10]	0.59364
J48 Consolidated	[1-11]	0.66852
Bagging - J48 Consolidated	[1-12]	0.71263
BayesNet	[1-13]	0.70531
K-NN	[1-15]	0.64216
BayesNet	[1-15]	0.72521
J48	[1-15]	0.73527
Stacking - Bagging	[1-15]	0.76207
Bagging - J48 Consolidated	[1-15]	0.78488

The main goal was to discover the Weka software and its classify function. Adding features improved the global accuracy. Indeed, the J48 algorithm increased its score by 0.14163 with 5 more features, the K-NN by 0.08061 and the Bagging by 0.07225 adding 3 features. It was good but not enough to reach a score near of 1 (100%). Moreover, increasing the number of features augments the time complexity of the algorithm and makes the classification process longer. It could also leads to errors due to irrelevant features for a precise case. To sum up, this is not a sustainable solution: I can not add new features infinitely.

5.2 Attribute selection tests

Weka provides a attribute selection function. It permits to select the most appropriate attributes for the classification, the most useful ones. Different algorithms has been tested. This is a list of the top of the podium, according to different rankings: Word Density, Number of characters, Confidence, Irony, Score Tag, Agreement.

A new series of tests has been made, using these 6 attributes only, for the training and testing files. The table 3 is regrouping some of the test results.

Table 3. Classification tests for 6 attributes

Classifier	Features used	Kaggle result
BayesNet	[1,2,3,4,13,14]	0.71087
K-NN	[1,2,3,4,13,14]	0.70140
J48	[1,2,3,4,13,14]	0.67422
Bagging - J48 Consolidated	[1,2,3,4,13,14]	0.73837

For J48 algorithm, it is important to notice that with 6 attributes, the score increased by 0.08058 in comparison to the test with 10 (using sentiments and emotions). I can suppose that most of sentiment and emotion attributes are not appropriate to the spam email classification. However, it is still not enough to reach the 0.73527 obtained with 15 attributes. On the four presented tests, only K-NN algorithm obtained a better results with 6 attributes instead of 15. To conclude on this point, feature selection is not adapted in this case.

5.3 Classification tests with more attributes

At this moment of the project, I was stuck. I did not know what to do to boost my score and I already have tested most of the classifiers. So I decided to add 3 relevant attributes: the number of links, of unique links and the number of words containing letters only [16,17,18]. Meanwhile, I also used a class balancer algorithm as pre-processing, because my dataset was imbalanced. I made new classifications with a total of 18 attributes and the table 4 present some results.

Table 4. Classification tests for 18 attributes, with class balancer

Classifier	Features used	Kaggle result
J48	[1-18]	0.79245
LWL	[1-18]	0.80676
Bagging - J48	[1-18]	0.81634
Vote - 5 classifiers	[1-18]	0.82456

Once again, results are a little bit better. For instance, J48 increased by 0.05718 compared to the test with 15 attributes. The score of 0.82456 has been reached using Vote with 5 classifiers: Bagging, Metacost, BayesNetTree, REP-Tree and J48. It applies a classification with all of these methods and choose the class predicted by the majority. The problem with this classification is that it is very long to process the file, and results are not much more better than another faster classifier.

5.4 The final solution

Throughout this project, many methods have been used to classify emails. Nonetheless, it has not been possible to obtain a better accuracy than 82%. Adding new features was not a viable option, as time complexity would increase and results would not be so accurate as expecting. Using new methods on existing features neither: many classifiers and pre-processing algorithms have been tested and none succeeded to show satisfying results. In order to improve the system, something had to be changed; something that would modify the whole classification process: the features. Indeed, tests dealt with numeric and nominal attributes, but the text in itself has never been used as a feature. Yet this could be the best asset, because the entire content is analyzed to classify an email. In this way, all the words usually used by spam are detected and the system can be much more accurate.

Therefore, only the text attribute has been used. Weka provides a method to transform words into vectors, and another to tokenize strings. That are the ones used in this series of tests, presented in table 5. Also, a class balancer pre-processing has been applied.

Table 5. Classification tests for the text attribute, with class balancer

Classifier	Kaggle result
Filtered classifier: String To Word Vector then J48 Consolidated	0.94817
Filtered classifier: String To Word Vector then SMO	0.96019
Naïve Bayes Multinomial text with default settings	0.97783

The Naïve Bayes Multinomial text algorithm reached an accuracy of 97.783%. This result is much more satisfying and acceptable than all other tests proposed in this paper. Moreover, the only feature needed was the text attribute, so the time spent on the text processing is minor. Focusing on this method, Table 6 presents tests for the Naïve Bayes Multinomial text algorithm with different settings.

For this classification, the best tokenizer is the default one: wordTokenizer. Reducing or increasing the word frequency is not good, as the score is, in both cases, lower than with the default frequency (3). Stemmers are also decreasing the accuracy. Therefore, The more accurate configuration is with default settings and adding the rainbow stop words handler, with. 97.888% of instances correctly classified, i.e 4035 instances over 4135. A 10-folds cross validation has been applied, to verify the score provided by Kaggle. The model has been build in 0.36 seconds and counts 17,012 different words. The Kaggle competition uses

Table 6. Naïve Bayes Multinomial text tests, with class balancer

Classifier	Kaggle result
Default settings	0.97783
Tokenizer: alphabeticTokenizer	0.97451
Tokenizer: nGramTokenizer	0.97645
WordFrequency: 2	0.96890
WordFrequency: 4	0.97173
Stemmer: lovinStemmer	0.97592
Stemmer: iteratedLovinStemmer	0.97365
StopWordsHandler: rainbow	0.97888
Stemmer: lovinStemmer, StopWordsHandler: rainbow	0.97854

the AUC evaluation metric, generally utilized for class imbalance problems.

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2} = \frac{1 + 0.978 - 0.022}{2} = 0.978 \quad (2)$$

6 CONCLUSION

In this paper, I built a system based on text content able to detect spam emails. I used a model and a testing dataset from the Kaggle competition "Detecting spam emails", created by the Dr. Octavio Loyola-González, for the Intelligent Systems course at Tecnologico de Monterrey, campus Puebla. I also used the Weka software for the classification. My first tests were dealing with the classification of emails using text analysis features. I added several attributes and tried many methods and classifiers. However, the results were not very accurate. The maximum score obtained, using 18 features, was 0.82456, which is not enough precise to be considered as a real detection system. Something was wrong with my model. So I reconsidered the basis of the classification: the features. I decided to make a classification only based on the whole text content by itself. I tried various classifier that can achieve it by transforming a string into evaluable values, such as word tokenizer or word to vector. This process was much more accurate, because all the words are analyzed and treated as features. In this way, the classifier can know that a spam is generally using this kind of words instead of others. The most accurate model was using the Naïve Bayes Multinomial Text classifier, with the Rainbow stop words handler and all other default settings. It is a fast and easy algorithm, based on probabilities. The system classified 97.888% of the 1036 instances correctly.

To conclude, spam email detection has been implemented for a long time on all email services. It is very important to filter emails for the user comfort, for his privacy and his security. However, these systems are not perfect as spam methods are always changing to bypass filtering. This paper showed that it is still very easy to make a content-based classification system, with a very good accuracy. It is important to understand that it has some limits: the dataset needs

to be updated and kept up to date, spammers could imitate real user emails, etc. This system could be improved combining it with other approaches, such as protocol-based or header-based.

References

1. Spam traffic share, Statista, 2020.
   https://www.statista.com/statistics/420391/spam-email-traffic-share/
2. Number of total email users in 2019, Statista, 2020.
   https://www.statista.com/statistics/255080/number-of-e-mail-users-worldwide/
3. Email definition, Wikipedia, 2020.
   https://en.wikipedia.org/wiki/Email/
4. Spam email definition, Wikipedia, 2020.
   https://en.wikipedia.org/wiki/Email_spam/
5. Email history, Wikipedia, 2020.
   https://en.wikipedia.org/wiki/History_of_email_spam
6. number of spam in 2014, MAAWG, 2014
   https://www.m3aawg.org/for-the-industry/email-metrics-report
7. Web spam, Crazyegg, 2018.
   https://www.crazyegg.com/blog/glossary/web-spam/
8. Types of spam, Malwarebytes.
   https://www.malwarebytes.com/spam/
9. SPIT in VoIP system, 2009.
   https://engineering.purdue.edu/dcsl/publications/papers/2009/
   spam-detection-dsn2009-rev-v13.pdf/
10. Detecting Spammers and Content Promoters in Video Social Networks, 2010.
    https://www.researchgate.net/publication/221299582_Detecting_Spammers_
    and_Content_Promoters_in_Online_Video_Social_Networks/
11. A Bayesian Approach to Filtering Junk E-Mail, 1998.
    https://www.aaai.org/Papers/Workshops/1998/WS-98-05/WS98-05-009.pdf/
12. Dont Follow Me - Spam detection in Twitter, 2010.
    https://www.researchgate.net/publication/221436356_Don't_Follow_Me_-_
    Spam_Detection_in_Twitter/
13. Combating Web Spam with TrustRank, 2004.
    http://www.vldb.org/conf/2004/RS15P3.PDF/
14. Kaggle competition for Intelligent Systems, 2020.
    https://www.kaggle.com/c/detecting-spam-emails/