TABLE OF CONTENTS

I. Introduction and Motivation

E-mails has became one of the most pupular and frequently used ways of communication, due to its worldwide accessibility, relatively fast message transfer, and low sending cost.

E-mail filters are commonly used to organise incoming mails, and remove spam mails and computer viruses, a less common use of these filters is to inspect outgoing mails, to ensure that for example, employees comply with appropriate laws in a companie.

Email spam, also known as junk email, are unsolicited mails that an email address can receive, it can contain some publicity, phishing, or unreal offers.

The number or spam mails has grown since the early 90s, and has becomming to include also harmful scripts and malwares that can infect our computer if the user open it.

This is the cisco statistics of spams for the last 18 months, we can clearly see that the amount of spams is huge, and almost equivalent to the total number of mails sent on internet :

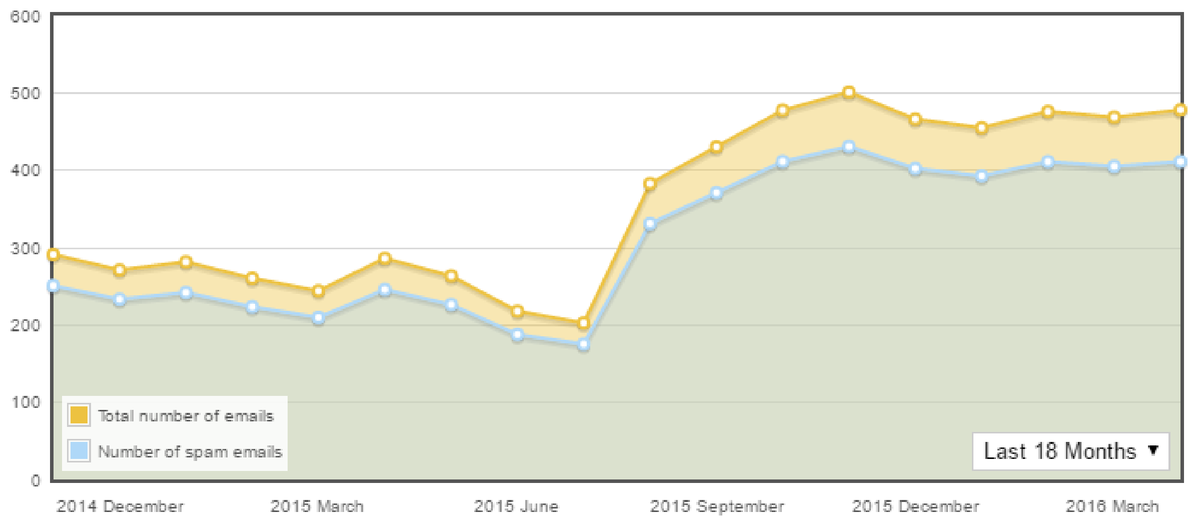


Fig 1: Cisco statistics

Spam mails is a major problem in today’s internet use, it can bring financial damage to companies, and annoys individual users.

II. Dataset

Having as objective to create spam filters using data mining approaches, i had to collect mails, so i started my work by looking for an open source dataset of spam and ham mails, and classify them in a way to have a training and testing mails :

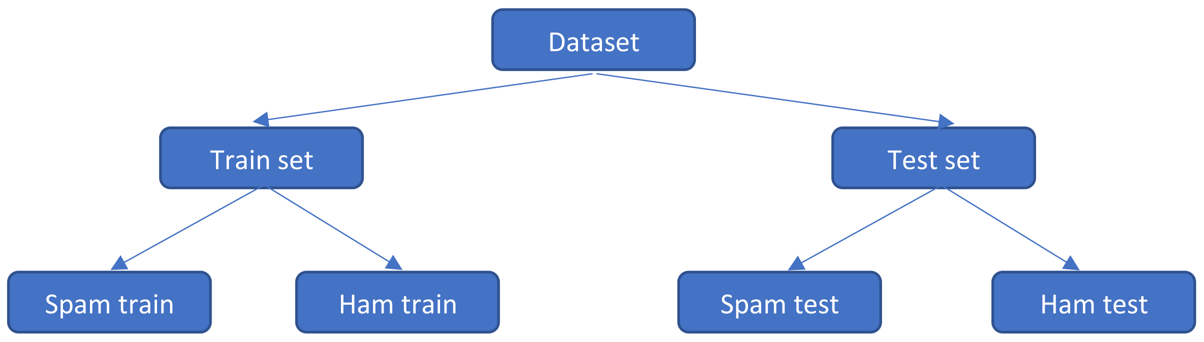


Fig 2: Dataset organisation

The dataset i worked on is an open source one, available on https://www.cs.cmu.edu, collected by 150 researchers, organized into forders, and contains a total of about 0.5 million mails. I organized it into three datasets, each one with different amount of mails :

DATASET	SPAM-TRAIN	HAM-TRAIN	SPAM-TEST	HAM-TEST	TOTAL
Dataset1	350	350	130	130	960
Dataset2	1000	1000	500	500	3000
Dataset3	1000	1000	400	400	2800

The dataset collection didn't stop at this stage, i actually reorganised my dataset, and downloaded new one as much as the work progressed.

III. PREPROCESSING

Before working on the mails, we had to preprocess the dataset messages. This processing has been devided into three steps :

- **Stop word removal :** Link words and some neutral ones had to be removed, like « the », « of », « and », because they are very common in english, and they can not help deciding whether a mail is spam or ham.
- **Lemmatization :** Words of the same family has being adjusted so that they all have the noun form.
- **Removal of non-words** : Numbers had been removed, and ponctuation replaced with spaces. All words in the email had been converted to lower case.

IV. TOKENIZATION METHOD

1. Introduction :

This method consists on taking each mail, and extracting all the words used, and from those words, and depending on the training done, the classifier will say if this mail is a spam or a ham.

2. Training :

Given the training set of spam and ham mails, we extracted all the tokens used in each of the training sets, and count its occurency in the spam, and ham train.

Denote :

W : Word (Token)
T(w/h) : The occurency of the token « w » in the ham train set.
T(w/s) : The occurency of the token « w » in the spam train set.

We can define the **spamicity** of a token :

$$S(w) = \frac{T(w/s)}{T(w/s)+T(w/h)}$$

Equation 1

The spamicity of each mail is calculated from the training set, the precision of spamicity is 105.

Threshold selection :

Each mail has a spamicity which is calculated by summing the spamicity of it's tokens after the preprocessing process :

$$S(mail) = \sum S(tokens)$$

Equation 2

We runned the spamicity program calculation on the spam and ham training set, and we obtained mails spamicity, that we represented like below :

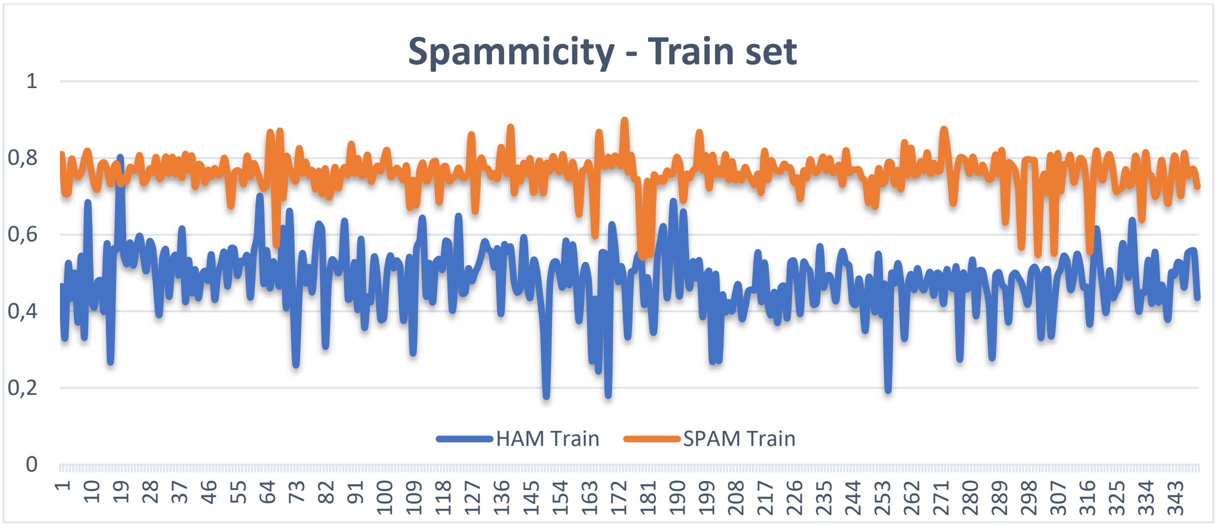


Fig 3: Spammicity - train set

We can clearly see that the spam mails have generally spammicity almost above 0,7, and the ham ones below 0,7, the problem is that we now need to know the best threshold, so we have the lowest error rate possible.

From this graph, we can see that the best threshold is located between 0,6 and 0,7.

In order to select exactly the best threshold, we tested all the values between in this area [0,6 ;0,7] automatically, with a precision at the order of 105, we obtained the following graph :

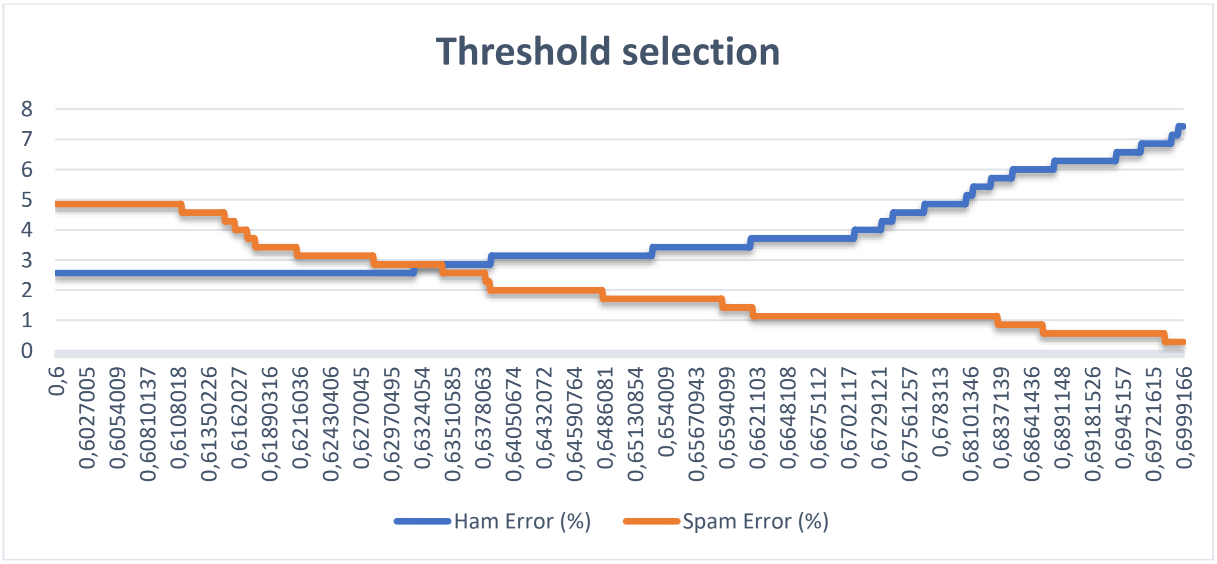


Fig 4: Threshold selection

The objectif of a spam filter is to block spams, but it also don't have to block ham emails, those ham emails can be very important emails, that the user can miss, and this error rate has to be the lowest possible.

In order to do that, we can choose the false positif error as the minimum possible, which matches 0.6294049.

Threshold = 0,6294049

3. Overlab :

After extracting the tokens used in spam and those used in ham mails, we create tokens table for each mail category (ham and spam) which contains the words used in each one.

The overlab is the number of tokens that are found in both ham and spam tokens table.

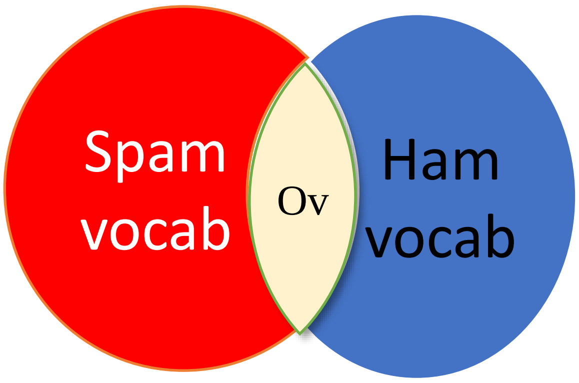


Fig 5: Overlab

The overlab is calculated by the following formula :

$$\text{Ov(HamTree, SpamTree)} = \frac{\text{size(overlap)}}{\text{size(spam tree)}+\text{size(ham tree)}-\text{size(overlap)}}$$

Equation 3

The overlab is the criterion to determine if a dataset is good or not, if the overlab is big, that means that there is a big amount of tokens that are used in both ham and spam mails, it is then not suitable to use a classifier based on this tokenization method.

4. Testing :

The dataset we are working on in this testing phase has a set of 700 mails for test, which contains 350 spams spam-test, and 350 hams ham-test.

We have another dataset containing 135 535 emails, we named it spamsDataset2 which are all spams.

We runned the filter on these datasets, with the treshold selected before, the results obtained are showed below :

DATASET	SPAM-TRAIN	HAM-TRAIN	SPAM-TEST	HAM-TEST	SPAMSDATASET2
Spams rate	97,42857%	2,857143%	96,92308%	4,6153846%	90,03456%
Hams rate	2,5714264%	97,142857%	3,0769196%	95,3846154%	9,96544%
Error rate	2,5714264%	2,857143%	3,0769196%	4,6153846%	9,96544%

5. <u>Conclusion :</u>

The spam filter depends on the training dataset used to train it, the more it is rich of selectif tokens, the more the error rate will be low, but the spams evolve too, they use actually pictures of texts, so that the tokens can not be detected with regular methods, they also alter some letters in the words so that they can not be considered as spam tokens, that's why the spam filter should be training dataset should be updated, and the spam filter trained regularly, to ensure having the lowest error rate.

From the results shown above, and with after the comparaison with other methods results, we can say that the tokens method used in my project is a good method to filter and protect an email.

V. NGRAM METHOD

Definition :

An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n − 1)–order Markov model. N-gram models are now widely used in probability, computational linguistics (for instance, statistical natural language processing), and data compression.

Two benefits of n-gram models (and algorithms that use them) are simplicity and scalability – with larger n, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently.

N-Gram viewer – google :

The Ngram Viewer was initially based on Google Books, but then switched to the 2009 edition of the Google Books Ngram Corpus.

The Google Ngram Viewer or Google Books Ngram Viewer is an online search engine that charts frequencies of any set of comma-delimited search strings using a yearly count of n-grams found in sources printed between 1500 and 2016 in Google's text corpora in numerous languages.

The program can search for a single word or a phrase, including misspellings or gibberish.

Dataset :

For training and testing my spam filter, we used several datasets, that we splited into train and test parts, the details are below :

- First dataset :
    - 350 spam train mails
    - 350 ham train mails
    - 130 spam test mails
    - 130 ham test mails
- Second dataset :
    - 4 000 spam train mails
    - 4 000 ham train mails
    - 500 spam test mails
    - 500 ham test mails
- Third dataset :
    - 1 000 spam train mails
    - 1 000 ham train mails
    - 400 spam test mails
    - 400 ham test mails

VI. TRAINING (PROBABILITY METHOD)

Text study:

Now that we have a dataset our dataset cleaned, each mail contains now the words used, the method we used has been inspired from the google n-gram model, but modified, the algorithm is as follow, given a « n », the mail text will be decomposed to « n » and « n-1 »grams, this will give us a spam and ham tokens table, and depending on the « n » used, the overlap between them will vary (the more the « n » is bigger, the lower the overlap is).

The training done, we obtain a set of spam and ham tokens and their occurrences.

Classifier :

In order to classifie the mails, we used the probabilities based method, we compute the probability of each token to belong either to spam or ham dictionary, the probability is calculated like below :

$$P\left({}^{Spam}/_{Gram}\right) = \frac{P\left({}^{Gram}/_{Spam}\right).P(Spam)}{P\left({}^{Gram}/_{Spam}\right).P(Spam) + P\left({}^{Gram}/_{Ham}\right).P(Ham)}$$

Where :

- $P\left({}^{Spam}/_{Gram}\right)$: Probability that a mail can be a spam, knowing that it contains that gram.

- $P\left({Gram}/{Spam}\right)$: Probability that this gram appears in a spam mail.
- $P({Gram}/{Ham})$: Probability that this gram appears in a ham mail.
- $P(Spam)$: Probability that a message can be a spam.
- $P(Ham)$: Probability that a message can be a ham.

We do the same for ham, to compute $P({Ham}/{Gram})$

In order to compute the probability of an mail to be a ham or spam, we sum the probabilities of all it grams :

$$P\ spam = \sum_{k=0}^{n} \mathrm{P}({Spam}/{Gram})$$

$$P\ ham = \sum_{k=0}^{n} \mathrm{P}({Ham}/{Gram})$$

We compare those two values to decide weither a mail is a spam or a ham.

VII. TESTING (PROBABILITY METHOD)

Results :

DATASET	N-GRAM	OVERLAP (%)	SPAM-TEST ERROR (%)	HAM-TEST ERROR(%)
Dataset1	6	18	3.7142868	2.2857132
Dataset2	13	2	2.0	0.5
Dataset3	15	3	0.099998474	0.0

The best result obtaines between all the datasets is 0% error rate, and the maximum is 3,7% error rate.

Analysis :

The n-gram choice depends on each dataset, because each dataset has it own words caracteristics, which means that the hamp and spam dictionary , and the average words length, are different.

The length of the ham and spam token tables depend on the n-gram chosen, and on each dataset.

DATASET	N-GRAM	OVERLAP (%)	SPAM TREE SIZE	HAM TREE SIZE
Dataset1	6	18	193808	264616
Dataset2	13	2	830081	412484
Dataset3	15	3	650403	808984

The overlap between the spam and ham tree is useful to say if a dataset is good or not, in other terms, it represents the amout of common tokens between spam and ham trees. The smaller the overlap is, the easier the distinctness between spam and ham mails.

The more the vocabulary table is big, the more the training stage is efficient, that means that there is less chances to be faced to an unseen token (who isn't in the table) during the testing stage, the unseen tokens in this method have 0 as probability of apprearence in spam and ham mails.

VIII. Training (threshold method)

Method :

In order to classify our dataset, we used in a second time a spammicity calculation method, we calculate the spammicity of each n-gram (token) in a mail, get their average, and have the spammicity of each mail.

Classifier :

The spammicity of each token is defined by the following formula :

$$S(w) = \frac{T(w/s)}{T(w/s)+T(w/h)}$$

$T(w/s)$: The number of appearence of the token in the spam mails.
$T(w/h)$: The number of appearence of the token in the ham mails.

The average of the spammicity in an e-mail is obtained by summing the spammicity of each token, deviding it by the number of tokens in the mail :

$$S(mail) = \frac{\sum S(tokens)}{N}$$

N : The number of tokens in a mail.

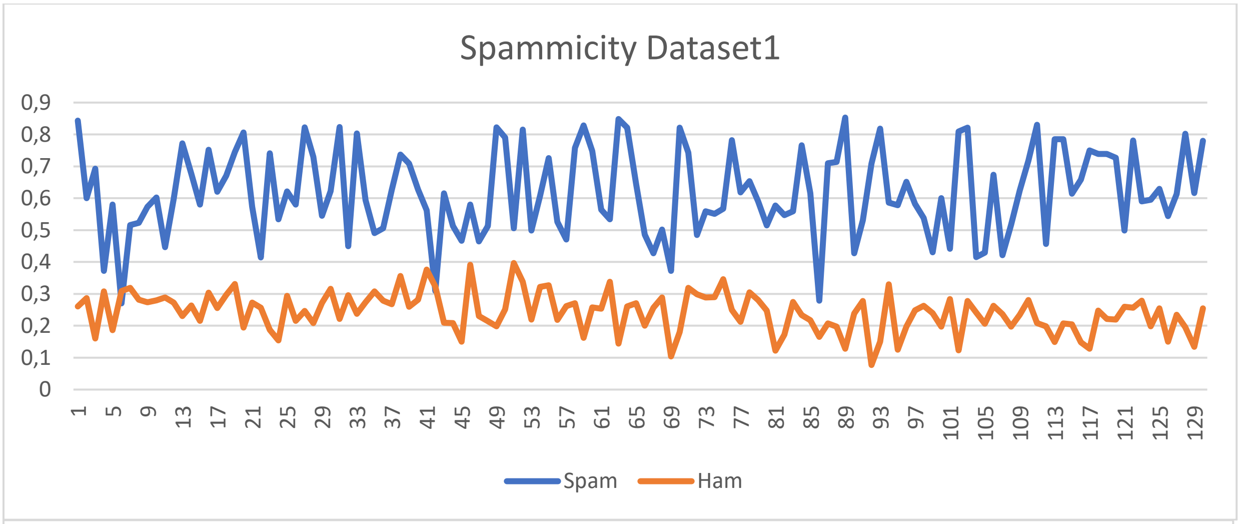
IX. Testing (Threshold method)

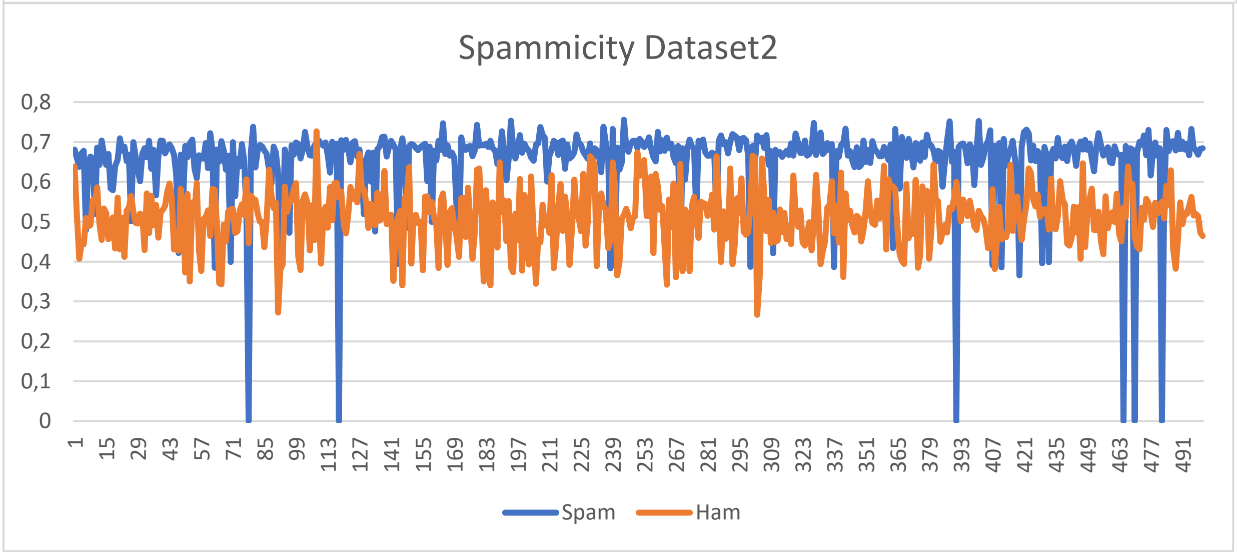
Threshold:

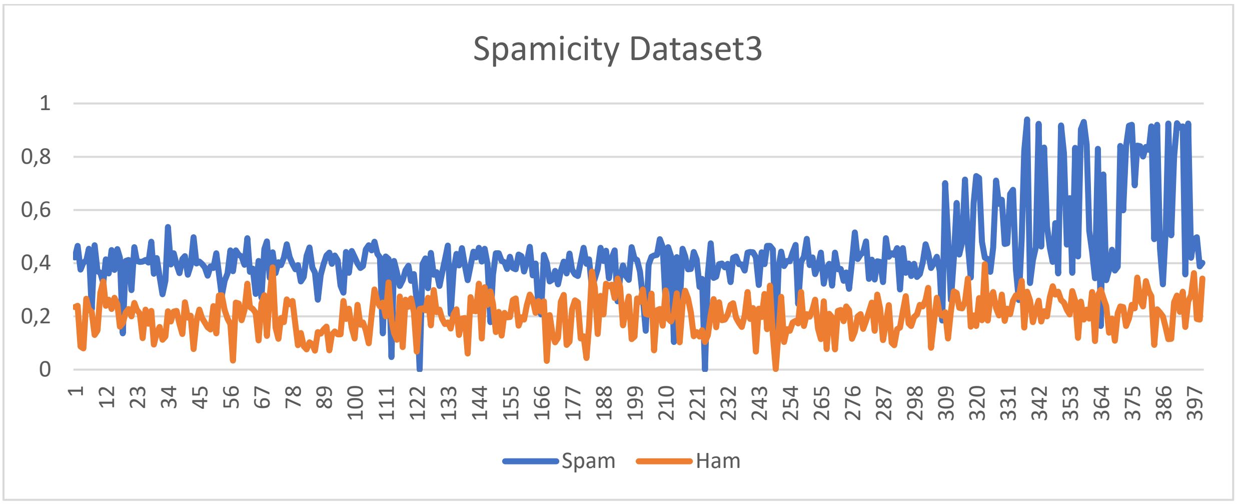
The threshold is a spammicity value that separates spam mails from ham ones. A threshold can be different from a dataset to annother, because every dataset can represent a user, who wants to configure his spam filter depending on the mails he is receiving.

Threshold selection :

Concerning every dataset we are working on, we calculate the spammicity in both ham and spam tests, plot them in a graph, to see the best threshold to choose, with a certain N-Gram constant to have the less error rate possible.







We can see a noticable difference between the tree graphics, in the first one, the difference between the ham and the spam functions is obvious, it is less in the second and the third one, this is due to the fact that the first dataset has an overlab less than the second and the third one, results are shown later.

Results :

After a deep analysis of multiple cases of thresholds with different n-grams, we obtained the following results :

DATASET	N-GRAM	TRESHOLD	SPAM-TEST ERROR (%)	HAM-TEST ERROR(%)
Dataset1	7	0.36999995	2.3076935%	2.3076935%
Dataset2	4	0.5999997	9.400002%	9.800003%
Dataset3	8	0.30999997	8,5	5.0

The best results are obtained in dataset1, the results depend on each dataset, some test datasets is very different from the train one, which means that there are a lot or unseen words in test mails, that haven't been seen in the train phase.

Analysing :

The n-gram constant chosen depends on the dataset, and the tokens that it contains, the more this value is big, the more the tokens table is big. The threshold chosen also depends on this n-gram value.

For the dataset we are working on, the result above correspond to this tokens dictionary values :

DATASET	N-GRAM	OVERLAP (%)	SPAM TREE SIZE	HAM TREE SIZE

Dataset1	7	13	310316	407022
Dataset2	4	24	26394	63196
Dataset3	8	14	354817	332557

The overlaps obtained are acceptable, and the size of the tokens tree too, our program conplexity is optimised, the runtime on the biggest dataset we have is 6,583021658 seconds, this is due to the methods and the collections structures too, TreeMaps are fast in writing and reading, that's why it has been used, because we have to write in the tokens dictionary tree everytime we have a new train mail, and update the number of occurrences in it, then we have to read in both spam and ham trees everytime we have a token, to check wether it is a new token or not, and calculate it's probability of appearence to get it's spammicity.

Conclusion

Comparing the results obtains with this method, with the one obtaines with the previous one, we can clearly see that the performance has been improved.

This n-gram adapted method is a very efficient, because it allows us, to find miss spelled words, a technic which is commonly used by spam bots, to get around standard spam filters.

The next step in my project will be to expand my method to unseen tokens, better than giving them 0 or 0,5 as probability. I will also try to extract features from my datasets, to use them on machine learning algorithms.

X. Neural network :

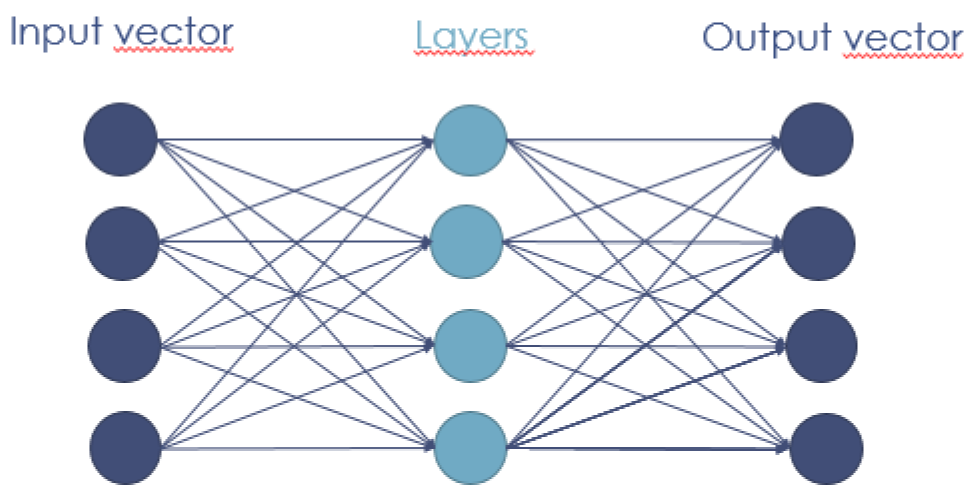
Definition

A **neural network**, also know as artificial neural network (ANN) is inspired by the human neural network, it is a function that is trained to estimate or approximate functions that depends on a large number of inputs that are generally unknown.

Perceptron algorithm :

The **perceptron** is an algorithm for supervised learning of binary classifiers, in other words, it is a function that can decide whether an input (represented by a vector of numbers) belongs to one class or to another.

Perceptron classifier



After collecting the ham and spam vocabulary and their occurrences, we create a table of random values, so that each random value is attributed to a token.

We compute then the product vector with the following equation :

*Product vector = TokenValue * Rand(token)*

We considere that a spam a positive product vector and a ham a negative one. In case this isn't, we have to regulate our random numbers so that our consideration is always true.

In case the spam product vector is negative, we change the token values according to this equation :

TokenValues = TokenValue + eta (0- output) * rand(token)*

In case the ham product vector is positive, we also change the token values according to this equation :

TokenValues = TokenValue + eta(1-output) * rand(token)*

Conclusion neural network

For extremely slow learning rate and low number of iterations, the classifier does not converge.

The best result is 96% on the third dataset with:

- 0.4 learning rate
- 120 iteration

XI. CLASSIFIERS COMBINATION

Motivation :

The results found with the previous classifiers are satisfiying, but each classifier has its advantages and disadvantages. To benefit from all the classifiers, we have to combine them, this is what i have done in this part.

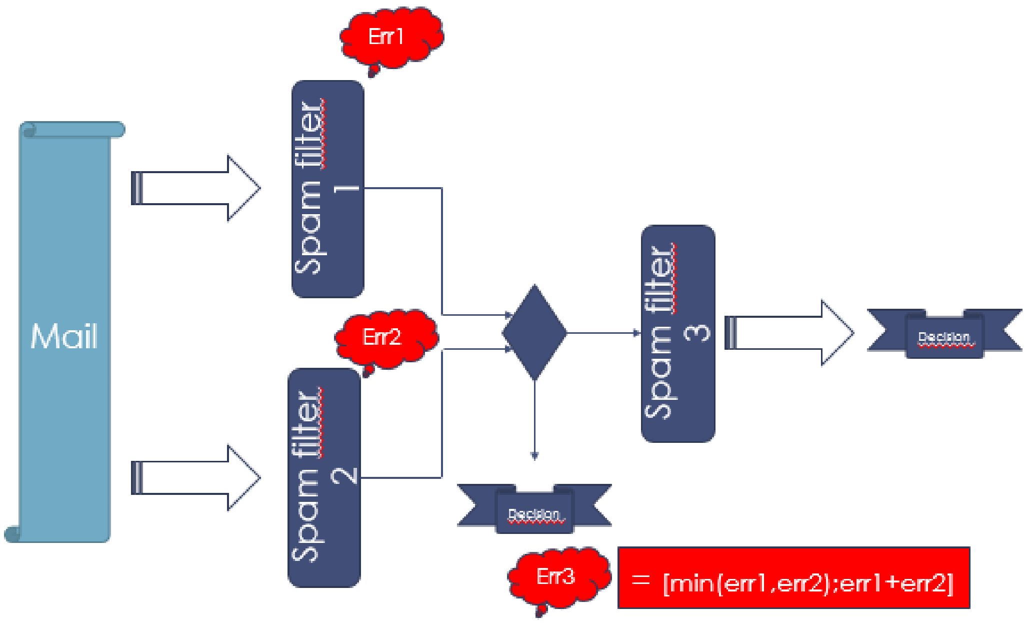


Fig 6: Classifiers combination illustration

Lets say that a we have a first mail to test, if the spam filter 1 and 2 said the same decision, this decision is the last one, if in the opposit, they said different decision, the mail is scanned by the spam filter 3, and it's decision is the last.

The error rate obtained with this combianation is between min(err1,err2) and err1+err2.

Results

For this classifier, we used just the first dataset.
Here are the results obtained after the use of the first and second classifier :

CLASSIFIER	SPAM ERROR RATE (%)	HAM ERROR RATE (%)	SPAM SAME DECISION	HAM SAME DECISION
Classifier 1	1.4	8.6		
Classifier 2	1.6	1.6	487	465

Now, the rest of the mails, the ones that the classifier one and two didn't give the same decision, will go to the third classifier, the following results aren't given with % because counting numbers are more significant in this case :

CLASSIFIER	SPAM LEFT	HAM LEFT	NUMBER SPAMS	NUMBER HAMS
Classifier 3	13	35	4	17

The spam/ham left is the number of mails that the classifier one and two didn't give the same decision, they are the ones re scanned by the third classifier, and the number of spams/hams is the number of spams/hams found.

FINAL RESULT	SPAM ERROR (%)	HAM ERROR (%)	NB CORRECT SPAMS	NB CORRECT HAMS	TOTAL NB ERROR	TOTAL ERROR RATE (%)
	1.8	3.6	491/500	482/500	21/1000	2.7

The error rate is related to the organisation of the classifiers, in the first case, we demonstrate that if we put the best classifier (the one who has the less error rate) in the first, the final error rate is bigger than if we put it at last (as the third classifier).

This graffic summarize the result obtained :

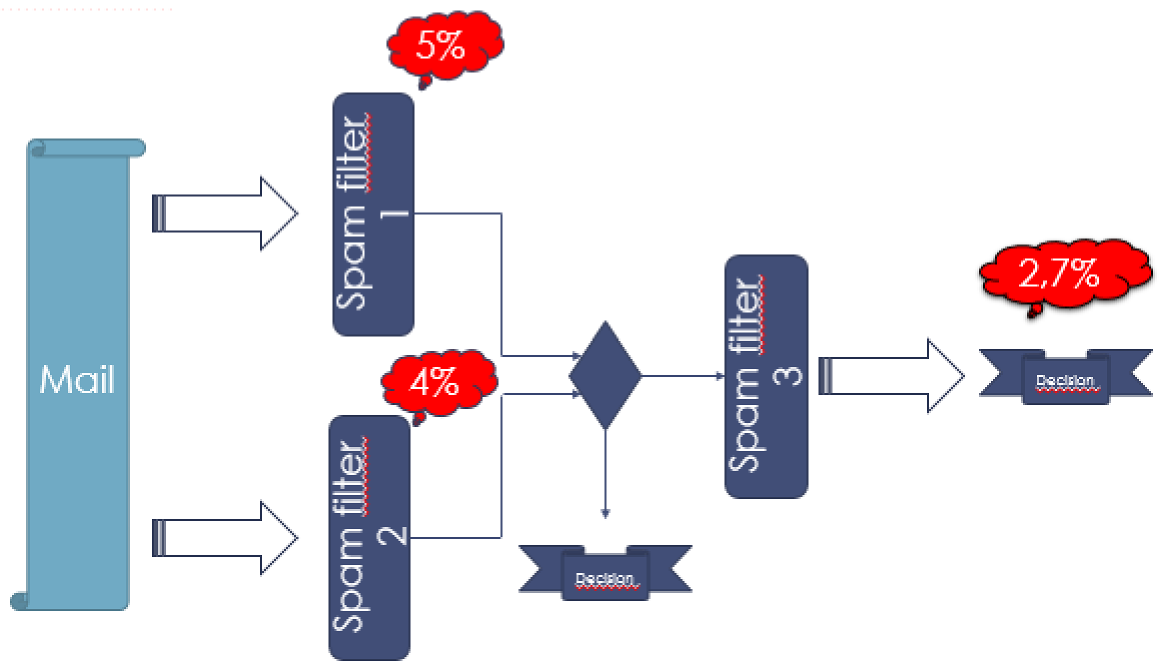


Fig 7: Classifiers first combination

When we reorganize the classifiers, here is the result obtained :

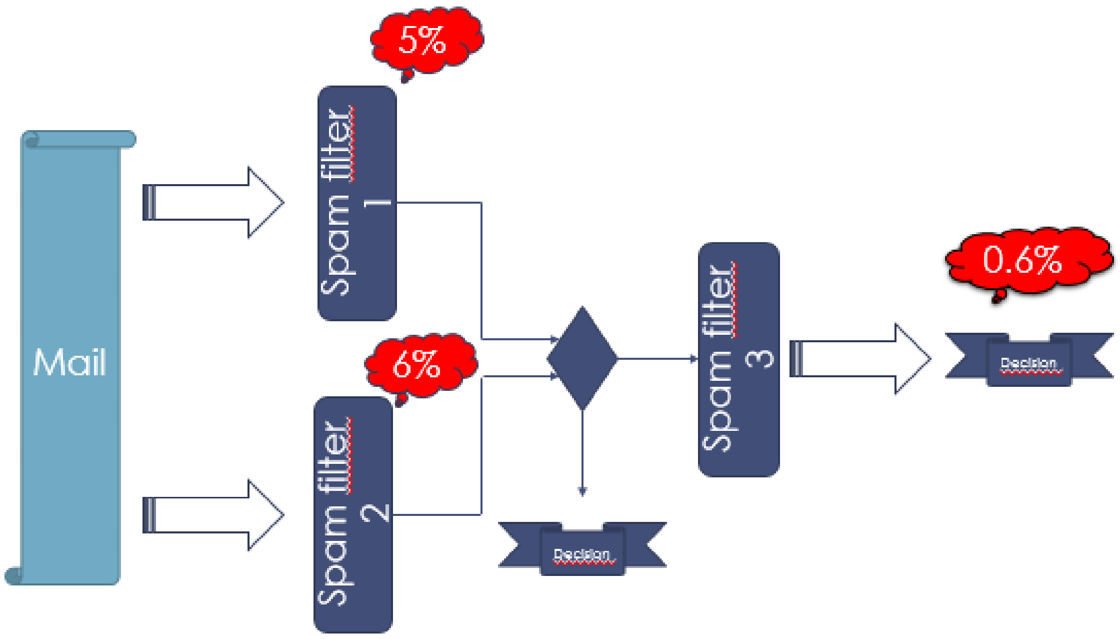


Fig 8: Classifiers second combination

Conclusion :

This classifiers combianation is the method that gives different approaches to a mail, it also benefits from all spam filters advantages, to reach the optimal performances, and it has less coutring methods.