Filtering spam at e-mail server level with improved CRM114

Víctor Méndez, Julio Cesar Hernandez, Jesus Carretero, Felix García

Department of Computer Science
Universidad Carlos III de Madrid

Abstract. Security managers and network engineers are increasingly required to implant corporative spam-filtering services. End-users don't want to interact with spam-classify applications, so network engineers usually have to implement and manage the spam-filtering system at the e-mail server. Due to the processing speeds needed to put these solutions into work at the server level, the options at hand are reduced to applications of the black-list/white-list type. This is the reason behind the fact that most applications based on AI techniques run only on the client side, particularly those based in the Naïve Bayes scheme, which has proved to be one of the most successful approaches to fight against spam, but nowadays is not as fast as other techniques and still not able to process the high amount of email traffic expected at a mail server. However, spam mutates and the spamies techniques have quickly evolved to easily pass the traditional black/white list applications, so there is a compelling need for the use of more advanced techniques at the server level, notably those based in the Naïve Bayes algorithm. This article explores this possibility and concludes that, simple improvements to a well-known Naïve-Bayes technique (CRM114[2]), following some ideas suggested in [8], could turn this algorithm into a much faster and significantly better one that, due to these improvements in speed, could be used at the server level.

1 Introduction

The problem of automatically filtering unwanted email messages is one of increasing importance, since bulk emailers take advantage of the great popularity of the electronic mail communication channel for indiscriminately flooding email accounts with unwanted advertisements. There are many factors which contribute to the proliferation of unsolicited spam, specially the inexpensive cost of sending email[14], and of obtaining pseudonyms[15]. On the other hand, we have the high cost associated with users receiving spam[2] and the network overflow.

The spam filtering problem can be seen as a particularly instance of a Text Categorization problem where the classes are Spam or Ham. In recent years, a vast

Méndez V., Cesar Hernandez J., Carretero J. and García F. (2004).
Filtering spam at e-mail server level with improved CRM114.
In *Proceedings of the 2nd International Workshop on Security in Information Systems*, pages 207-216
DOI: 10.5220/0002674702070216

amount of techniques have been applied to solve this problem. Some of the top-performing methods are Learning Rules that Classify e-mail[20](1996) based in the RIPPER algorithm, Ensembles of Decision Trees[16](1999), Support Vector Machines[17](1998), Boosting[18](2000), and Instance Based Learning[19](2000). Nowadays, advanced Naive Bayes methods are the top-performing ones, coming from Paul Graham principles for spam-classifying [3], and some basic improvements [4](2003). The false positives go from 0,3% to 0,06%, and the detected spam from 99,5% to 99,75%[4].

On this paper we are going to explain principles that makes CRM114 one of the best accuracy filtering application. We after compare the design features and obtained results with other state-of-the-art applications, and expose our approach to the problem modifying CRM114 behavior on window size for empirically comparing accuracy versus speed, and on features text extraction, introducing the concept of virtual feature, that will finally modify the original SBPH polinomy used on CRM114. We present two experiments results and we extract some conclusions.

2 Sparse Binary Polynomial Hashing (SBPH) CRM114

SBPH creates a lot of distinctive features from an incoming text, then the Naive Bayes technique is applied to this features instead of directly to the words. For this purpose the algorithm slides a window of length five words over the incoming text and, for each window state, generates a set of order-preserving sub-phrases containing combinations of these words. These order-preserved sub-phrases are processed calculating 32-bit hashes, and with all the resulting sub-phrases the algorithm creates a 32-bit superhash (i.e. hashing of hashes) value that will be used to calculate the Naive Bayes probabilities. Essentially, each sub-phrase tries to extract a word feature from the text. With a window size of five words, each word affects 2^{5-1}=16 features.

The performance of CRM114 Mailfilter from Nov 1 to Dec 1, 2002: 0.068% of false negatives and ZERO false positives[2]. Its filtering speed on classification is less than 20Kbytes per second (on a Transmeta 666 MHz) [2], which obviously hinders the use of crm114 for filtering at the mail-server. For this purpose, an average network needs a classification speed of at least 60kb/sg [13].

3 CRM-114 and other state of the art filtering applications

Every application has distinctive characteristics that we summarize on the next diagram, where we show the features for some state of the art antispam applications.

Table 1.

	1	*2*	*3*	*4*	*5*	*6*	*7*	***False positives***
Crm114 [2]	Yes	5	S	Yes	GPL	M, S (?)	crm	0.00% [2]
SpamAssesine [6]	Yes	Unknown	A	Yes	No	M, I	Unknown	0.19% [6]
Gnus-emacs [7]	Unknown	No	N	No	GPL	M(+IMAP)	gnus	Unknown
SpamProbe [8]	Yes	1,2	S	Yes	QPL	P(+fetchmail)	No	0.035% [8]
PopFile [9]	Yes	Unknown	A	Yes	GPL	M, P.	PERL	0.125% [10]

1.Automatic featured extractor from text
2.Phrase window size.
3.HTML filter: No HTML filter (N), Simple HTML filter (S), Advanced HTML filter process (A)
4.Specific design to arise with false-positives.
5.Software license.
6.Filtering level at the: MTA Client (M), POP3 Proxy (P), At the e-mail server (S) Internet level [6](I).
7.Generic filter language [2].

All the applications are based on the naive-bayes algorithm, except SpamAssassin, which is based on a combination of a GA and rules, which is probably the reason of the worse false positives rate. The false positives rates are taken after different learn cycles, depending on the application approach to the optimum value. We can see this data as a kind of "how good can it do it". The classifying experiments were made with a different number of mails; every author has used different sets, but always in the order of thousands. It is clear from the table above that crm114 has the better false positive rate.

Automatic featured extractor for text is very important for the detection of new spamies techniques [5], which simplify the network engineer task of coping with new tactics. It is also known that both crm114 and SpamProbe use a similar windowed word philosophy, which will be explained below.

All the applications except gnus-emacs use some type of HTML processing. This is becoming important due to the fact that a lot of spamies techniques are based on HTML use [9]. We can also see that gnus-emacs has not implicit design to manage false positives.

Additionally, some kind of free software license is needed to give the network engineer the possibility of escalating or updating the code, or for tuning in a production domain without the strategic dependency on a specific software developer. If not a completely open code license, at least a specific generic filter language that allows for some level of implementation-specific tuning should be offered. This is especially important on spam-classify applications because they generally don't have good generalization features but are able to successfully operate in a real world domain after only small design changes. From our point of view, the main drawback of SpamAssassin for our purpose is that it has not free software license or an open generic filter language, so it works outsourcing the spam-filter at internet level, a solution that is not appropriate for a corporative implementation at the server level. On the other hand, crm114 could filter at both the MTA Client level and also at the

server level, but only if we could greatly increase its classification speed. The rest of applications could run only on the client side.

4 The window philosophy

The crm114 algorithm uses a window size of five words. Most researches like Brian Burton [8] indicate that window sizes over two words generally produce no better accuracy or false positives rates, and in fact may well be worse because they could lead to an overflow of features and, additionally, they greatly decrease the filtering speed. It is obvious that crm114, which is a combination of an advanced Naive Bayes method and polynomial hashing, has a very different window philosophy: bigger windows gives polynomial techniques a more relevant weight in the final combined method. If we set the window size to two words, we will use a two variables polinomy that is less suitable than a bigger polinomy for feature extraction. The first consideration we have to do is that following the crm114 principles for relationship between window size and features, for a two words windows size we may extract only 2 features, and this sounded a little too poor. So at the end of the day, we think in trying different empirical experiments playing with both window size and word features. For this purpose we converted the static crm114 compiler into a dynamic matrix of pipelines (window size) and superhashes phrases (number of words features) to help on false positives service level and classify rate decisions.

5 The virtual features

How are we going to extract different number of features than SBHP features relationship with the window size, in order of 2^{N-1} **?** We are not going to follow such relationship, for example with 2 words window size we repeat original SBHP coefficients patterns, $2^{2-1} = 2$ patterns until 20, and we call them virtual features.
Depending on the conjunction of window size and virtual features, we will obtain diferents SBHP functions, taken as algorithmical seed the original Yerazunis function with 5 x 16 dimension.

6 Test I

6.1 Benchmark corpus

Our benchmark corpus contains the learning mails set to create the .css files, (superhashes mapped files). Yerazunis recommends a learning corpus around 0,5 Mb size, and following his recommendation we have used the file nonspamtext.txt (695111 bytes extracted from my personal inbox and public mail lists asfsdevel, SI-edu, or SL-admin) and the file spamtext.txt (536547 bytes from a public set [11]).

The classify mails set to test our approach come from individuals donations [12], thus the learn/classify sources are independent enough to generalize results at the mail server filter level. Following Paul Graham indications, they approximately have the same ham/spam distribution (ham=170, spam=220)

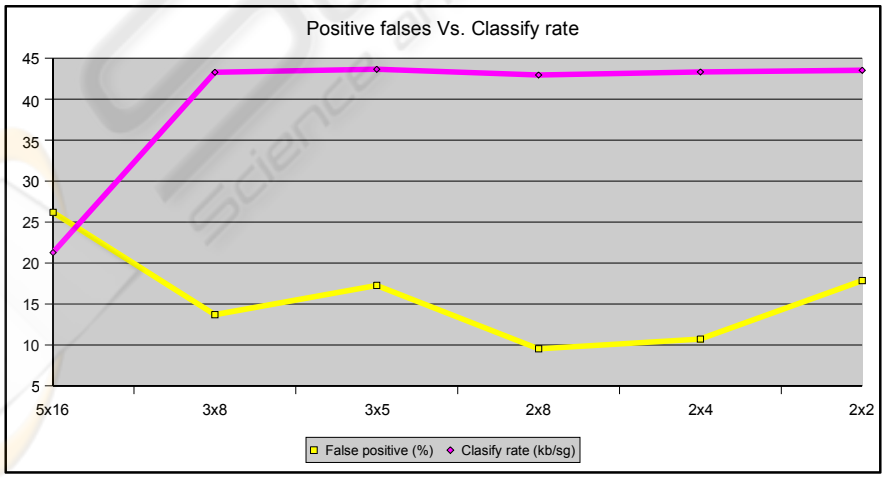
6.2 Results

The tests show the result for single-pass learning, without any retraining cycle, so results do not shown a "how good can it do it", but are enough for comparison between different matrix sizes. The test were done on an 700Mhz. Intel Pentium III Copermine (c) with 128Mb memory, and a processor load over 99% for the classify process. The times are taken strictly on classify process part. The matrix size are relative to pipelines(window size) x superhashes phrases.

Table 2.

	5x12	3x8	3x5	2x8	2x4	2x2
Spam detected	87,61%	88,53%	94,95%	92,20%	92,66%	95,87%
False positive	26,19%	13,69%	17,26%	9,52%	10,71%	17,86%
sg. training spam	496,67	376,86	374,05	373,34	373,6	374,98
Sg training ham	36,91	27,99	26,37	25,82	26,27	25,75
Training rate kb/sg	2,25	2,97	3	3,01	3,01	3
Classify spam	50,48	24,33	24,32	24,32	24,52	24,41
Classify ham	62,61	31,25	30,81	31,7	31,01	30,87
Classify rate kb/sg	21,27	43,28	43,64	42,95	43,32	43,53

Fig. 1.



The graph above show the values for the critical parameters. This confirms worse false positive rates if the window size is over two words. Remember the original static crm114 matrix size is 5x16, so crm114 has similar behavior on the window size value to the Brian Burton study for the SpamProbe[8], which obtains betters results on 1 and 2 words window sizes. We also obtain much better classify speed rates but this was an obvious result because, for example, the original crm114 has to calculate 5x16=80 hash for a superhash feature, and, at the best false positive performance only 2x8=16 hash are to be computed for a superhash feature.

6.3 Test I Conclusions

We have proposed a new approach for implementing spam filtering on the email server which is a modification and also an improvement over the state-of-the-art crm114 technique and leads to much higher speeds, due to the fact that it uses a window word size of only two words and, surprisingly enough, also to better classification and false positive rates.

7 Test II

Now we would like see how our approach works with a bigger test corpus, in order of thousands mails from SpamAssasine public corpus[21]. We also play with relearning cycles, following our aim of ZERO false positives, that original crm114 may accurate[2], and checking if our conclusions for a one cycle of Test I are also valid in a real domain making maps proccess. For this purpose we first train up to medium size corpus, we after relearn every false classificated case up to final corpus size, with bigger ham corpus than spam one, trying to force more ham weight on maps, for better results on false positives. We finally test from a different corpus for statatistics. We will combine in a natural way the .css maps and the test corpus, for example easy ham map and spam map, testing with easy ham and spam corpus(EASY-EASY); and we also check the cross map-corpus tests, for example easy maps with hard test corpus, that are not expected to get good results, but we want to check it.

7.1 Benchmark corpus II

We focus our study on 2 words window size and original crn114 5x16 matrix for comparison.
On the first two phases we take mails from 2003 SpamAssasine public corpus, which has a singular ham sources classification with easy to classify ham, and hard ham that usually produces a worse false positives rate, so we are going to test with a ham map done with easy ham, and other with a mix of easy and hard ham on first phase, and only hard ham on relearn phase, we will call it mix-ham, but is some kind of hard ham with little enough easy ham. After relearn classifying thousands mails, we obtain map files of the following corpus size and mails number:

Table 3.

	2x2		*2x4*		*2x8*		*2x12*		*2x16*		*5x16*	
Mail class	***Bytes***	***Mails***	***Bytes***	***Mails***	***Bytes***	***Mails***	***Bytes***	***Mails***	***Bytes***	***Mails***	***Bytes***	***mails***
spam	349107	67	349317	65	331767	60	314745	54	332300	50	308988	67
Easy-ham	508550	62	500116	62	498545	62	500256	62	405325	63	292499	60
mix-ham	414676	48	419989	46	677018	46	673904	44	653895	41	416095	53

Training corpus: Spam Assasin public corpus 2002. Mails:

Spam: 501

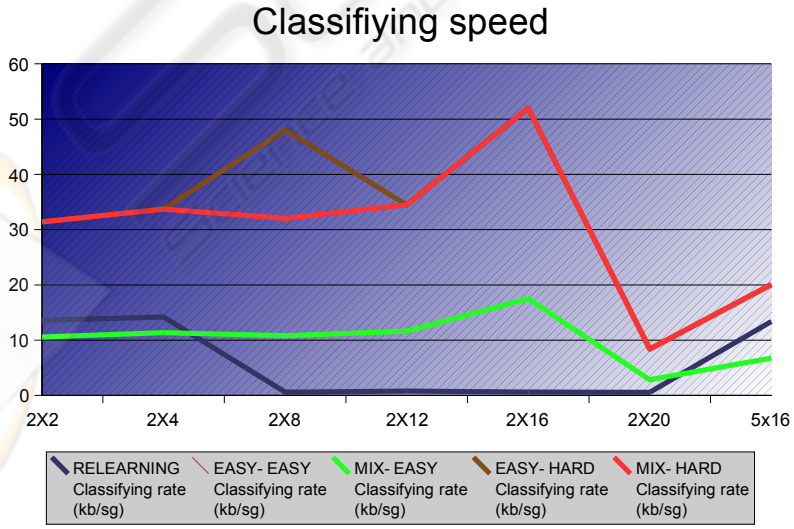
Easy ham: 2551

Hard ham: 250

7.2 Test II Results

The test were done on an 700Mhz. Intel Pentium III Copermine (c) with 128Mb memory, and a processor load over 99% for the classify process. The times are taken strictly on classify process part. The matrix size are relative to pipelines(window size) x superhashes phrases.

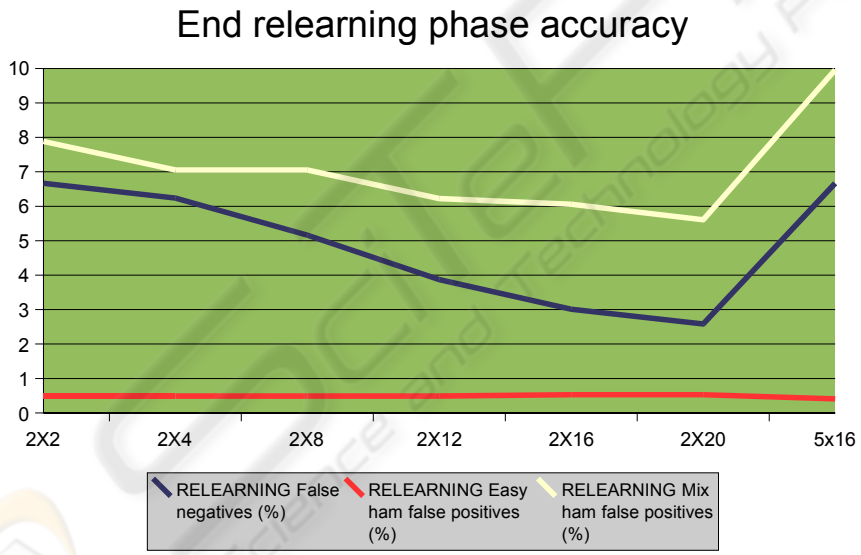
Fig. 2.



The classify speed shown on the graph above, confirm similar conclusions than Test I. The relearning classify rate has better results on original 5x16 matrix, than some of the two words window size matrix. But this data are not relevant because the critical classify speed is on production time, not in relearning phase. On the other hand we can see the testing classify rates, working better with 2 words window size, specially with 12 and 16 features, and with the bests results on tests that uses hard ham corpus. We also can see better results on original 5x16 matrix than in the 2x20, so 2x20 will be out of consideration because speed deficiencies.

For the next graph we have to do the consideration that during relearning phase the maps are changing, training the maps with every false classificated case of the test. So the accuracy will also change and the data we shown are taken from the beginning to the end of relearning phase, for comparison with the test phase accuracy(see below).

Fig. 3.



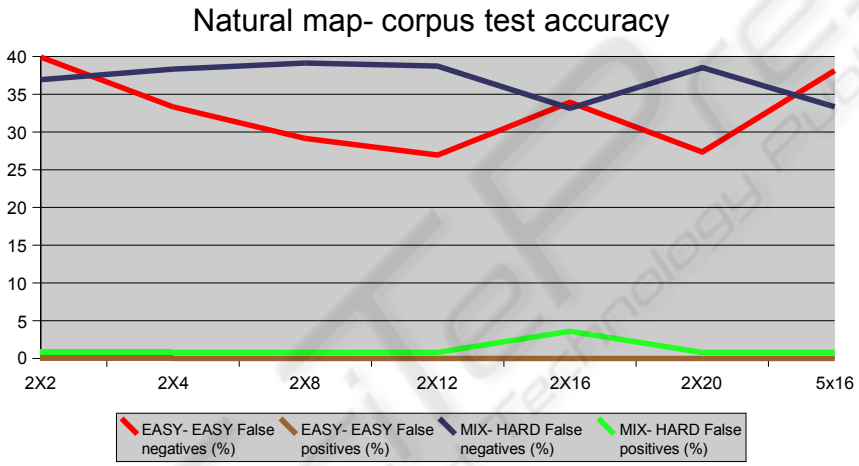
We confirm better results on two words window size, and much better with more features. We also confirm the easy and hard ham SpamAssasine classification, that goes for false positives from around 0,5 % value for easy ham, to more than 6% with hard ham.

Diagram below shown the accuracy for natural tests: one is the spam and easy ham maps versus spam and easy ham test corpus, the other is spam and mix ham maps versus spam and hard ham corpus.

We get the ZERO false positives for easy ham case. The best results are in 2x12 and 2x20 but we have to remember that 2x20 has the big problem of speed. The green and brown lines are the false positives and our previous working thesis of "more weight on ham maps for better false positives results" obtain here empirical confirmation, if

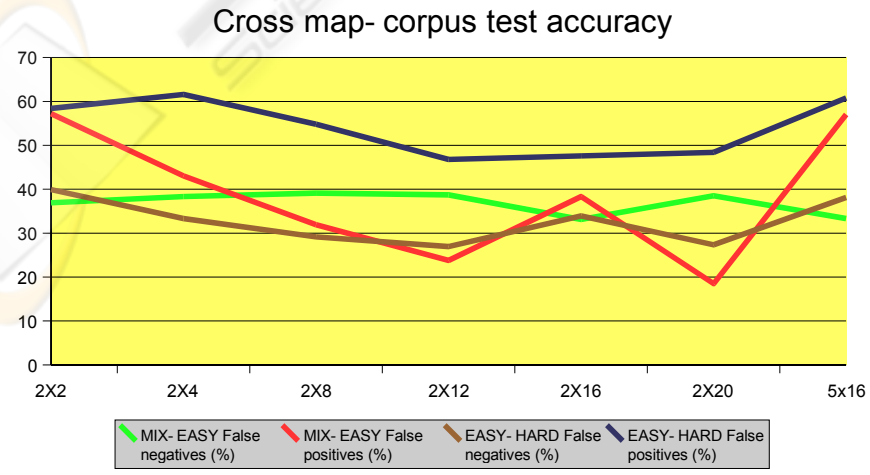
we compare with relearning accuracy, where the maps at beginning were of similar size. But in the other hand we have very bad results on false negatives. We can clearly see this dependency in the 2x16, where mix-hard false negatives decrease if we compare with other matrix size, but false positives increases. The important fact for our study is that we can diminish false positives playing with learning and relearning final mails corpus size, but increasing the false negatives. We may tuning for a agreement solution to obtain also ZERO false positives with not so bad false negatives.

Fig. 4.



The next diagram shown that not natural map-corpus combination are not good to work.

Fig. 5.



Conclusions

After the more reliable Test II, we confirm the conclusions of Test I, our approach has better accuracy and speed than original 5x16 static matrix size crm114. We also extract more conclusions, the most important is that the ZERO false positives are within reach if we use relearning on false cases in a planned way, and we have proposed a valid tactic in two phases for this aim. An other conclusion is we can give more important to the market critical parameter "false positives", using bigger ham corpus size than the spam one, to make the maps (.css); but this decrease the other false cases, the negative, the spam and ham maps (and they train corpus) are an antinomy with absolute dependencies one to each other for the final results. We also observed that a ham subdivision in hard an easy may be good for planing the train and relearning tactic, we have to considerer that hard ham is not very used, but are those mails that could be easy mistaken with spam, even for the human eye, so depending on corporative domain we should specially train the maps for them, or not. Finally we have shown that not natural map-working domain, are not valid at all.

References

[1] Tom M. Mitchell. Machine Learning - McGraw-Hill, ISBN: 0-07-042807-7

[2] William S. Yerazunis. Sparse Binary Polynomial Hashing and the CRM114 Discriminator - MER Labs. Cambridge, MA. 2003 and Cambridge Spam Conference Proceeding - http://crm114.sourceforge.net/

[3] Paul Graham. A Plan for Spam. 2003 Cambridge Spam Conference Proceeding http://paulgraham.com/spam.html

[4] Paul Graham. Better Bayesian Filtering. 2003 Cambridge Spam Conference Proceeding http://paulgraham.com/better.html

[5] Jason D.M. Rennie, y Tommie Jaakkola. Automatic Featured Induction for Text Classification. - MIT, AI Labs. Abstract Book. 2002 and 2003 Spam Conference- http://www.ai.mit.edu/~jrennie/spamconference/

[6] Matt Sergeant. Internet Level Spam Detection and SpamAssassin 2.50.- 2003 Cambridge Spam Conference Proceeding - http://axkit.org/docs/presentations/spam/

[7] Teodor Zlatanov. Spam Analisys in Gnus with spam. - 2003 Cambridge Spam Conference Proceeding - http://lifelogs.com/spam/spam.html

[8] Brian Burton. SpamProbe: Bayesian Spam Filtering Tweaks - 2003 Cambridge Spam Conference Proceeding - http://spamprobe.sourceforge.net/index.html

[9] John Graham The spammers compendium.- 2003 Cambridge Spam Conference Proceeding - http://popfile.sourceforge.net

[10] Kristian Eide. Winning the War on spam: Comparison of Bayesian spam filters. 2003. http://home.dataparty.no/kristian/reviews/bayesian/

[11] Unam public spam set 2002-2003: http://www.seguridad.unam.mx/Servicios/spam/spam/

[12] From call for donations for this specific use, at the Universidad Carlos III de Madrid, 2003.

[13] Personal communication with Juan Carlos Martin, Security and Network Manager of EspacioIT, which has over 3.000 mail users along different domains and mail servers. October, 2003

[14] Carreras & Marquez. Boosting Trees for Antispam Email Filtering. 2001 TLAP Research Center. LSI Department. Universitat Politecnica de Catalunya.

[15] L.F. Cranor and B.A. LaMaochia. Spam Comunications of the ACM, 1998.
[16] Sholan M.Weiss and others. Maximizing text-mining performance. - 1999 IEEE Intelligents Systems.-
[17] Joachims. Text categorization with support vector machine. Proc. 10th Eur. Conf. Machine Learning. 1998.
[18] R.E. Schapire and Y.Singer. BoosText: boosting based system for categorization Machine Learning. 2000.
[19] Yang & Liu. A re-examination of text categorization methods. Proc. 22nd ADM SIGIR Conference 1999.
[20] W.Cohen Learning Rules for Classifying Mail. AAAI Spring Symposium on Machine Learning in Information Access. 1996.
[21] http://spamassassin.org/publiccorpus/ . - public - corpus AT jmason dot org if you have questions.