Seven Hypothesis about Spam Filtering

The CRM114 Team

Correspondence Author:
William S. Yerazunis, PhD
Mitsubishi Electric Research Laboratories
wsy@merl.com

Abstract: For TREC 2006, the CRM114 team considered several different hypothesis on the topic of spam filtering. The hypothesis were that:

1 Spammers were changing tactics to successfully evade content-based spam filters;
2 A pretrained database of known spam and nonspam improves overall accuracy;
3 Repeated training methods are more effective than single-pass Train Only Errors training
4 KNN/Hyperspace classifiers are more effective than classical Bayesian or Markovian classifiers
5 Delaying feedback learning results in degraded filter accuracy
6 Bit-entropy filters are as good or better than tokenizing filters
and after-the-fact:
7 1-ROCA% is the best figure of merit for spam filters

Of these hypothesis, we found that spammers were not significantly able to evade content based spam filters, that pretraining is probably not helpful, that repeated-pass training is not significantly helpful, that KNNs are of roughly equal accuracy to computationally and storage-equivalent Markov classifiers, that delayed feedback is only marginal in impacting filter accuracy, and that despite their highly counterintuitive design, bit-entropic filters are capable of similar or better accuracy to tokenizing filters. We also found a fascinating counter-correllation between 1-ROCA% and the final accuracy of a filter (the accuracy of the filter for the final 10% of the corpus).

Introduction

This year, the CRM114 team was in the unique position of being able to test a number of hypotheses that have been of interest to the spam filtering and text categorization communities, in an "apples to apples" comparison. Because of CRM114's architecture, it is easy to change classifier designs without needing to revisit any of the backing mechanics of spam filtering, and a fairly reasonable library of inter-changeble (though generally incompatible) classifiers has built up. Since last year's TREC, CRM114 has added two new classifiers (a KNN / Hyperspace classifier and a bit-entropy classifier). Because these new classifiers use the same CRM114 "wrapper" we can isolate changes in the classifier versus changes in the training methods, with other variables (including, to significant extent, designer and programmer skill) held relatively constant.

Additionally, the full CRM114 system is available to anyone as a free, open-source, GPLed download; other researchers who want to further experiment with new corpora, new training methods, or even wholly new classifiers may download and modify the source code.

Rather than submit four completely new classifiers, we chose to submit one of our runs as a duplicate to last year's work, so as to be able to directly compare the "old" Mr. X corpora (prior to 2005) versus a the most recent corpus from the same source. The first configuration we submitted to TREC 2006 is essentially identical to our high performance thick-threshold-trained OSB classifier from TREC 2005 [Assis 2005] and provides an apples-to-apples benchmark of the multiple TREC Mr. X corpora. Specifically, this run deals with the question of whether spammers had indeed evolved their techniques to successfully evade content-based filters; if so, the performance of the filter should drop between the Mr. X 2005 and Mr X 2006 corpora.

For the second CRM114 run, we submitted a hopefully-improved learning algorithm. This included a tighter training tolerance and a complete, real life, up-to-date pretrained database containing about 200,000 example multi-word tokens. This database was the current, active filter database of one of the authors. Our goal in this run is to determine if pretrained databases are actually of any help, both in the initial high-error regime of the learning process, and if the initial learning is any aid or impediment to long term accuracy.

For the third CRM114 run, we considered the TREC 2005 curve set and noted that the CRM114 OSB classifier exhibited a strange dip in the curve, where the accuracy became very good, and then seemed to deteriorate, as though the filter was becoming overconfident. To prevent this loss of accuracy, we considered a very aggressive learning protocol where prior known texts are reexamined repeatedly and retrained if within the error bound. This required the filter to repeatedly access the prior texts; on average two old texts (of known category) were reexamined for each new text classified.

The fourth CRM114 run was intended to test a classifier based on KNN (K-Nearest-Neighbor) classification called "Hyperspace". [Yerazunis 2005]. The Hyperspace classifier is equivalent to a KNN classifier with an infinitely large neighborhood; the weight contribution of each neighbor is defined in terms of the weighted distance in a 4-billion-dimension hyperspace (this is also equivalent to a Parzen window of size equal to the learned corpus).

We also obtained an informal run of a bit-entropy classifier based on the overall concepts of Andrej Bratko et al of the Institut Josef Stefan, as described and tested in TREC 2005 [Bratko 2005]. Note that this bit-entropy classifier is wholly new code and was written based only on the published descriptions of the DMC algorithm (Cormack and Horspool, 1987) [Cormack 1987]. It does not contain any of the IJS code. This code also includes a novel method of growing the entropy-prediction model that does not yet include DMC-style node splitting, and with modifications to produce a system that can run with relatively small memory (less than 64 megabytes per class, without resetting) and O(constant) time for both classification and learning. This classifier is still under development and shouldn't be considered

“the last word” in bit-entropy classification by any means.

Hypothesis 1: Spammers are Successfully Adapting to Statistical Filters

This is probably false. It appears that spammers have not yet figured out how to penetrate user-taught statistical learning filters.

The evidence to support this negative result is that the "classic" CRM114 OSB-Markov model with only non-behavior-altering bugfixes from the TREC 2005 best practices runs, and shows undiminished effectiveness versus the "old" MrX1 and "new" MrX2 test sets. Assuming that the MrX2 test set covers the “recent” past sufficiently, there is no statistical evidence that spammers have adapted to successfully circumvent statistical filters. More explicitly, the 1-ROCA%s is 0.177 (90% confidence interval 0.128 to 0.246) for the old (TREC 2005) MrX test set versus the TREC 2006 Run 1 with 1-ROCA% of 0.1498 (confidence interval of 0.1051 to 0.2134).

It should be noted that the OSB-Markov model used for this test is a rather “advanced” model; in particular it does not use single-word tokens at all. Instead, adjacent words and short phrases with interior words omitted of lengths up to five words are the basic features used for this classifier. Such longer-phrase systems have now been adapted as default by other spam filters such as dSpam, and as an option on other systems, so the conclusion of immunity to adaptation by spammers is not a CRM114-only claim. This claim does not extend to simple single-word classifiers; although CRM114 has a single-word Bayesian mode, we didn’t run that experiment and hence have no useful data.

Hypothesis 2: Training Information is Portable from One User To Another

Unfortunately, this hypothesis is also probably false. Copying a baseline of learned statistics from one user to be used as initial state for a second user does not seem to improve final accuracy. However, the number of errors in the initial training phase can be decreased by about a factor of two by supplying pretraining based on the corpus of another user. Whether this is valuable in the marketing sense is not clear, but it seems that long-term accuracy is not improved, and in fact, it may be slightly damaged.

The evidence for dismissal of this hypothesis is a comparison of CRM114 Run 2 with CRM114 Run 1. CRM114 run 2 is the same executable as run 1 (a "classic" OSB-Markov configuration), but Run 2 carries an initial preload of one author’s personal statistical filtering criteria (plus a wider learning band and more storage to accomodate this preload). This preload contained approximately 200,000 multi-word features; about two-thirds of them are hapaxes (seen only once in the training set). Unfortunately, Run 2 is not statistically significantly better on any test set than Run 1; hence the preload did not help.

On the B2 test corpus, the untrained Run 1 (no pretrain) had a total of 99 errors, with a 1-ROCA% of 0.38 (confidence interval 0.2350 to 0.6309) while the pretrained Run 2 had 87 errors and a 1-ROCA%

of 0.4196 (confidence interval 0.2822 to 0.6235). This difference is insignificant.

On the X2 testset, Run 1 (no pretrain) had 430 total errors and a 1-ROCA% of 0.1498 (confidence interval 0.1051 to 0.2134), while the Run 2 (pretrained) had only 188 total errors but the 1-ROCA% was still insignificantly different at 0.1592 (confidence interval 0.1115 to 0.2271). The interesting result that the total errors fell significantly, but the 1-ROCA% was essentially constant, implying that the initial pretraining did not affect overall accuracy in either direction.

Therefore, we conclude that a preload of someone else's corpus does not help final accuracy, but may yield a faster "startup" in the learning process.

Hypothesis 3: Intensive Training on the Same Corpus Improves Accuracy

It is probably not the case that repeated training significantly improve accuracy. The evidence for this is that the CRM114 Run 3 is statistically not differentiated from CRM114 Run 1; Run 3 uses repeat training while Run 1 does not; both are identical OSB-Markov systems.

The tested protocol for repeat training as used in Run 3 is to behave identically to the thick threshold training from Run 1 up until corpus element 100 (out of the 45,000 elements of the MrX2 corpus), when the system switches to repeat training. At this point, after each "new" corpus element is tested and then considered for training, two previously known corpus elements are reconsidered for training as well. This "one new, then two old" training process repeats for the remainder of the corpus.

The actual comparison values for Run 3 (repeat training) on the X2 corpus were 279 total errors and a 1-ROCA% of 0.1393 (confidence interval 0.0973 to 0.1995) versus Run 1 (control) at 430 total errors and a 1-ROCA% of 0.1498 (confidence interval of 0.1051 to 0.2134). Against the B2 corpus Run 3 (repeat training) got 98 total errors and a 1-ROCA% of 0.2983 (confidence interval 0.1742 to 0.5105) while Run 1 (control) got 99 errors and a 1-ROCA% of 0.3852 (confidence interval 0.2350 to 0.6309) This is not significantly different; therefore we dismiss the hypothesis as unproven.

Hypothesis 4: KNN/Hyperspace Classification is Superior to Bayesian Classification

Sadly, this hypothesis is not supported by the evidence. CRM114 Run 4 did not show a statistically significant improvement in filtering, nor did an informal resubmit to correct the weighting formula to a more complicated but accurate one.

The algorithm used in the official Run 4 is a modified KNN with the neighborhood set to the entire previous corpus, and two different weightings. The first weighting is "similarity only", and uses the number of shared features as the inverse distance of the known and unknown texts; this was the officially submitted Run 4. Additionally, we obtained an "informal" second test run. The second test

weighting uses an inverse distance equal to the number of shared features squared, divided by the product of features seen in the known but not in the unknown times the number of features seen in the unknown but not the known (this is the recommended and default configuration for CRM114's Hyperspace classifier). In either weighting method, the inverse distance is used in a $1/R^2$summation to obtain the total weight for each class on the unknown text; the text is assigned to the class with the largest $1/R^2$sum.

The actual results for this are that the similarity-only inverse distance metric (official Run 4) on the MrX2 corpus yielded 667 errors and a 1-ROCA% of 0.3056 (confidence interval 0.2428 to 0.3846), and 387 errors and a 1-ROCA% of 0.9653 (confidence interval 0.7767 to 1.1992) on the B2 corpus. The similarity-squared over difference metric (informal run) yielded just 241 errors and a 1-ROCA% of 0.2217 (confidence interval 0.1682 to 0.2923), a far better showing. This is competitive with the CRM114 official Run 1 (OSB) with 420 errors and an overall 1-ROCA% of 0.1498 (confidence interval of 0.1051 to 0.2134).

Thus, we can feel relatively safe concluding that a similarity-squared over difference metric used in an infinite neighborhood KNN is probably competitive with other statistical methods, but not distinctly superior to it. Choice of the KNN versus a Bayesian or other filter should probably be made on the basis of experimentation against the live data being classified.

Hypothesis 5: Delaying Feedback Degrades Learning Filter Accuracy

Finally, some good news: it seems that delaying feedback to the filters degrades accuracy to a "barely significant" extent on some corpora but does not degrade accuracy significantly on others.

The evidence for this is based on the delayed corpora such as the MrX2 corpus; for Run 1 (classic CRM114 OSB) the relevant numbers are (undelayed) 420 errors and an overall 1-ROCA% of 0.1498 (confidence interval of 0.1051 to 0.2134). Adding a delay to the error feedback loop changes this to 972 total errors but 1-ROCA% only changes to 0.1341 (confidence interval 0.0953 to 0.1887); paradoxically the total errors increased but the 1-ROCA% figure of merit decreased.

Similarly, Run 2 (pretraining) changed only to 269 total errors and a 1-ROCA% of 0.1143 (confidence interval 0.0837 to 0.1561) which is not statistically significant. Run 3 (repeat training) moved to 465 total errors and a 1-ROCA% of 0.1129 (confidence interval 0.0873 to 0.1460) which is not statistically significant, and Run 4 (KNN/Hyperspace similarity-only metric) moved to 891 total errors and 1-ROCA% of 0.4898 (0.4230 to 0.5671). This final degradation is actually significant, but this is against the similarity-only metric which is known to be weak. Against the informal improved hyperspace, the values were 388 total errors and 1-ROCA% of 0.3518 (confidence interval 0.2962 – 0.4177), also barely significant.

Against the B2 corpus, Run 1 (no pretrain) had a total of 99 errors, with a 1-ROCA% of 0.38

(confidence interval 0.2350 to 0.6309) which moved to a delayed 145 errors and 1-ROCA% of 0.6346 (confidence interval 0.4395 to 0.9155). This move is moderately significant. Run 2 (pretraining, without delay) had 87 errors and a 1-ROCA% of 0.4196 (confidence interval 0.2822 to 0.6235); adding delay to the error feedback loop changes this to 191 total errors and 1-ROCA% of 0.6006 (0.4366 to 0.8257). This difference is at the edge of significance but not quite at the 90% level. Run 3 (agressive retraining) on the undelayed B2 corpus gives got 98 total errors and a 1-ROCA% of 0.2983 (confidence interval 0.1742 to 0.5105); adding delay to the error feedback loop yields 187 total errors and a 1-ROCA% of 0.4584 (0.3205 to 0.6551); another result on the edge of 90% significance. Against the known-weak similarity-only KNN of Run 4 the change was 387 total errors and a 1-ROCA% of 0.9653 (0.7767 to 1.1992) without delay, and 567 total errors and 1-ROCA% of 2.009 (1.7386 to 2.3018) with delay (a strongly significant result).

As a rough generalization, it seems that the better the classifier works to start with against a corpus, the better the classifier does when a delayed error feedback is imposed. It may also be that the effect of delay in training errors is magnified by small corpora (corpus B2 is only 12,000 elements versus X2 at 45,000 elements)

Hypothesis 6: Bit-Entropy Works Better than Statistical Filtering

Our final hypothesis was to test a bit-entropy filter against the corpora. As this would put us "over limit", for the official limit of four runs, we obtained informal runs after the official submissions had all completed (and thank the the Spam Track Chair for making this possible).

The bit-entropy classifier is based on Cormack-Horspool DMC compression, as adapted by Andrej Bratko of the Institut Josef Stefan (TREC 2005). However, the code tested was developed completely independently of the Cormack-Horspool and Bratko code. Additionally, the code does not use the DMC node-splitting nor David Young-Lai's node merging algorithm [Young-Lai 1999]. The basic principle is still the same – use an optimal compression algorithm as the kernel of a classifier. Each known corpus is used to dynamically build a compression model; the model which compresses the unknown text to the smallest number of bits is the model that best fits the unknown text. Unlike the Bratko algorithm, the CRM114 bit-entropy classifier does not actually calculate the best such encoding; it merely calculates the length of that best encoding.

The CRM114 bit-entropy classifier produces the compression model by starting from a single node, and then allocating and linking in a new node on each new bit of known text. The algorithm also an arithmetical-coded local prior history on each node already allocated; when the actual prior history of the bit pattern being encoded is within a very small threshold of a node already allocated, instead of a new node allocation, the current transition is linked to the already-existing node (this saves tremendous amounts of space; sensitivity testing confirms that it does not affect accuracy by more than 20% but brings the memory requirement down to about 64 megabytes at the "flat part" of the performance

curve; adding more memory does not seem to help past that limit, at least against our corpora). By arithmetical prior history coding and by maintaining a lookaside table to get very close to the correct coded node in a single lookup, the CRM114 bit-entropy classifier is able to run almost as fast as the classic CRM114 OSB-Markov classifier.

During classification we measure the information in each actual transition according to:

$$\text{entropy} = -\log_2(P_{\text{transition}})$$

The entropy of the entire text is merely the sum of the entropies of the individual bits. Note that this is not the average entropy of the source (that being the sum over all transitions of -P log2 P; the initial -P being the weighted probability of the transition ever occurring; we're posteriori in this case).

The bit-per-node runtime allocation does not guarantee closed bit-transition models, so unknown texts will often "run off the edge" of the model. In this case, the classifier needs to resume the model at an appropriate place; this node is located via the arithmetical-coding prior history system. In the case of an "off the edge" transition, we need to encode only the signal that the model was forced to jump to the best appropriate place; since both the encoder and decoder can calculate the identity of this best node trivially (at the point of the jump, both the encoder and decoder know the full prior history of the text), the number of bits required to encode this transition is only $-\log_2(1 / (N_{total}+1))$.

The results of this run are an independent confirmation of the IJS filter presented last year by Andrej Bratko as a viable classification system; despite it's highly counter-intuitive mode of operation (being completely blind to the concept of a "token", and always operating at the bit-at-a-time level), the filter works quite well. Additionally, this code operates well with less than 64 megabytes per class, which is a significant advance on the 2005 IJS code.

The actual results for the bit-entropy filter are 849 total errors and a 1-ROCA% of 0.1610 (confidence interval of 0.1215 to 0.2131) for the MrX2 corpus, and with delayed feedback this accuracy drops to 1887 errors and a 1-ROCA% of 0.4099 (0.3580 to 0.4693). This compares well against the CRM114 Run 1 of 430 total errors 1-ROCA% of 0.1498 (confidence interval of 0.1051 to 0.2134).

Additionally, against the public test corpora, the CRM114 bit-entropy classifier displays the same "flat-curve" ROC response that the IJS filter shows. This suggests that the "flat ROC" curve of the 2005 IJS filter may be an intrinsic effect of bit-entropic classifiers; thus, for situations where the cost of one kind or another is very large, the bit-entropy classifier may well perform better than a Bayesian classifier.

Other Metrics – Terminal Error Rate

There are other possible metrics for ranking filter accuracy beyond the error transfer matrix and the 1-

ROCA%. Of particular interest is the terminal error rate – the TER - that is, the error rate as seen by end user of an already-deployed system, which is defined as the error rate in the last 10% of the test corpus. As we hadn't set out to test this, it's not really a "testable hypothesis"; it's more of an "interesting observation of Nature" which warrants further critical examination.

Comparing TER versus 1-ROCA% yields the following interesting results for the 2006 TREC "public" test set of about 92,000 messages:

Classifier	*1-ROCA%*	*Rank By ROCA%*	*Terminal Error Rate*	*Rank by Terminal Error Rate*	*Discrepancy of Rank (ROCA rank – TER rank)*
OSB (Run 1)	0.1498 (.1051-.2134	**1 (tied 4 ways)**	5	6	-5
OSB Retraining (Run 3)	0.1353 (.0973 - .1995)	**1 (tied 4 ways)**	4	4 (tied 2 ways	-3
OSB-Pretrain (Run 2)	0.1592 (.1115 - .2271)	**1 (tied 4 ways)**	4	4 (tied 2 ways)	-3
Bit-Entropy (Informal run 2)	0.1610 (.1215 - .2131)	**1 (tied 4 ways)**	2	3	2
Hyperspace Similar/Difference (Informal run 1)	0.2277 (.1682-.2923)	5	1	**1 (tied 2 ways)**	4
Hyperspace Similarity (Run 4)	0.3056 (.2428-.3846)	6	1	**1 (tied 2 ways)**	5

The worst classifiers according to 1-ROCA% were the Hyperspace classifiers, but Hyperspace had only a single error in the last 9219 messages (the actual value was one error in the last 10,000 messages, for a TER of less than 0.01% and a terminal accuracy of better than 99.99%), while the "best" filters according to 1-ROCA% had five times as many errors.

It seems that good behavior in terms of 1-ROCA% may be correllated to a weak terminal error rate. If so, that means the filter gives a poor end-user experience when compared to a high 1-ROCA%, but otherwise very TER-accurate classifier (which would have 1/5th the error rate once trained).

Conclusions

We can express our TREC 2006 conclusions concisely in the following table, assuming that 1-ROCA% is a good figure of merit (we don't have TER values for the private MrX1, MrX2, or B2 corpora).

CRM114 TREC 2006 Conclusion Summary

Hypothesis	*Experimental Sample*	*Control Sample*	*Conclusion*
Spammers have adapted to statistical filters	2006 Run1: MrX2 430 total errors 1-ROCA% 0.1498 (.1051-.2134	2005 Run 3 MrX1 1-ROCA% 0.177 (0.128 – 0.246)	**N O** No significant difference between the old and new MrX corpora results.
Training is portable from one user to another	Run 2 MrX2 183 total errors 1-ROCA% 0.1592 (.1115 - .2271)	Run 1 MrX2 430 total errors 1-ROCA% 0.1498 (.1051-.2134)	**N O** Total errors are lower, but 1-ROCA% is worse
Intensive, repeated training improves accuracy	Run 3 MrX2 279 total errors 1-ROCA% 0.1353 (.0973 - .1995)	Run 1 MrX2 430 total errors 1-ROCA% 0.1498 (.1051-.2134)	**N O** Total errors are lower, but 1-ROCA% is not significantly better
KNN/Hyperspace is superior to Bayes/Markov	Run 4 MrX2 667 total errors 1-ROCA% 0.3056 (.2428-.3846) Informal 1 MrX2 241 total errors 1-ROCA% 0.2277 (.1682-.2923)	Run 1 MrX2 430 total errors 1-ROCA% 0.1498 (.1051-.2134)	**M A Y B E** The 1-ROCA% is not quite as good, but the lower total error count and <u>much</u> lower terminal error rate are "very interesting"
Delaying feedback degrades accuracy	Run 1 MrX2 Delay 972 total errors 1-ROCA% 0.1341 (.0953-.1887)	Run 1 MrX2 430 total errors 1-ROCA% 0.1498 (.1051-.2134)	**M A Y B E** The MrX2 corpus doesn't show significant degradation, but the B2 corpus shows a barely significant degradation at 90% certainty
Bit-entropy classification actually works	Informal 2 MrX2 849 total errors 1-ROCA% 0.1610 (.1215 - .2131	Run 1 MrX2 430 total errors 1-ROCA% 0.1498 (.1051-.2134)	**C O N F I R M E D** Overall 1-ROCA% is within the 90% statistical certainty bounds
1-ROCA% is the best figure of merit for filters	All runs ranked by terminal error rate (TER)	All runs ranked by 1-ROCA%	**M A Y B E N O T** TER anti-correlated with 1-ROCA% but positively correllated with the enduser experience

Thanks and Acknowledgements

The reader should note that to some extent or another, a very large group of people have submitted quality reports, anecdotal evidence, optimizations, bugfixes, and improvements to the CRM114 classification engines; to a person, they all decline to take public credit for it as "their part was too small to warrant credit or authorship". Among them are Fidelis Assis, Christian Seifkes, Paolo Pazzoli, and numerous others known only to the Correspondence Author as online identities.

In any case, the Correspondence Author wishes to publicly acknowledge their help. In whatever form or way they might have helped, the help was invaluable.

The Correspondence Author also wishes to specifically thank Gordon Cormack (the Track Chair) for great assistance in both the formal runs and the additional informal runs.

Bibliography

Assis 2005 - CRM114 versus Mr. X: CRM114 Notes for the TREC 2005 Spam Track, Assis, F., Yerazunis, W.S., Siefkes, C., Chhabra, S. NIST TREC Conference 2005

Bratko 2005 - Spam Filtering using Character-level Markov Models: Experiments for the TREC 2005 Spam Track, Bratko, A., Filipi˘1c, B, NIST TREC Conference 2005

Cormack 1987- Data Compression Using Dynamic Markov Modelling, G. V. Cormack and R. N. S. Horspool, The Computer Journal, Vol 30, No. 6, 1987

Yerazunis 2005 – Sorting Spam with K-Nearest-Neighbor and Hyperspace Classifiers, Yerazunis, W. S., Assis, F., Siefkes, C., Chhabra, S., MIT Spam Conference 2005

Young-Lai 1999 - Adding state merging to the DMC data compression algorithm, Young -Lai, M. Information Processing Letters 70 (1999) 223–228