Three Non-Bayesian Methods of Spam Filtration: CRM114 at TREC 2007

Mamoru Kato [1,2]
Joseph Langeway [1,4]
Yimin Wu [1,3]
William S. Yerazunis, PhD [1] *

1: Mitsubishi Electric Research Laboratories, Cambridge, MA
2: Mitsubishi Electric Information Technology Center, Ofuna, Japan
3: University of Massachusetts at Amherst, MA
4: Southern Connecticut State University

* Correspondence Author

Abstract:

For the TREC 2007 conference, the CRM114 team considered three non-Bayesian methods of spam filtration in the CRM114 framework – an SVM based on the "hyperspace" feature==document paradigm, a bit-entropy matcher, and substring compression based on LZ77. As a calibration yardstick, we used the well-tested and widely used CRM114 OSB markov random field system (basically unchanged since 2003). The results show that the SVM has a spam-filtering accuracy of about a factor of two to three better accuracy than the OSB system, that substring compression is somewhat more accurate than OSB, and that bit entropy is somewhat less accurate for the TREC 2007 test sets.

Introduction:

CRM114 is an open-source, GPLed language framework that makes it easy to construct arbitrary text modification and classification engines. The current version of CRM114 contains standard Bayesian classifiers, classifiers based on word parsing to create a Markov Random Field, orthogonal sparse bigram random fields (OSB) optionally enhanced with the EDDC confidence factor), Littlestone's "Winnow" algorithm (a linear perceptron), a K-nearest-neighbor ("Hyperspace") nonlinear classifier, a linear SVM operating in the KNN space, a bit-entropy classifier, a classifier based on Lempel-Ziv compression, and a three-layer neural network classifier with back-propagation training.

The TREC Spam Track allows teams to enter up to four spam classifiers for formal testing against the secret NIST database. For TREC 2007, we chose to formally test three of the new classifiers and

retain our “standard yardstick” of the 2003 OSB classifier as the fourth test. The three new classifiers selected for testing were the SVM operating in a one-feature-per-document space, an enhanced bit-entropy classifier (with a short-range “precognition” system enabled in the codec), and a classifier based on the LZ-77 text compression algorithm (Lempel and Ziv, 1977).

Additionally, we requested an informal run of a fourth experimental classifier based on a three-layer neural network with back-propagation, however the results of that run were not available at press time for this paper.

Testing Corpora:

The filters were tested against a battery of spam/nonspam corpora. For 2007, there were two basic corpora – a “public” test set with nonspam email from the ENRON corpus and recent spam, and a private (secret) corpus based on the 2007 emails of a Mr. X (believed to be a college professor but otherwise Mr. X's identity is a secret; the emails in the Mr. X corpus are never released).
The public corpus was submitted in four different forms:

1. an immediate form, which is an idealized case where each of the documents is classified in time sequence, and then the filter is given the correct answer and allowed to retrain itself on that document before going on to the next document;
2. a delayed form, where the filter is not given the correct answer until some other messages are classified (and possibly trained) first;
3. a partial form, where some subset of the messages are not allowed to be trained;
4. an active form, where the filter is given access to all of the message texts, and is allowed to ask for the correct types of a small subset of the messages and must then classify the other messages based on just those subset

The MrX corpus was tested with only the “immediate” and “delayed” form. We did not build a version for “active” testing, so we have no results in that area.

The Classifiers:

We tested four classifiers in the “official” runs. All four classifiers were tested in a SSTTT format (single-sided thick threshold training). In SSTTT training, the classifier runs against the text and returns a result. If the result is incorrect, the text is trained in. Additionally, if the result is correct, but not judged correct by a sufficiently large margin, then the text is trained in as well.

All four classifiers were trained in this way. No classifier received “negative” or “double-sided” training (although it might be considered that the two-class C- SVM implementation obtained negative training because calculation of the ideal hyperplane and support vector set requires knowledge of both

classes. The CRM114 SVM implementation is a single two-class SVM, not a series of 1-class SVMs.)

Classifier Details - OSB: Our benchmark “control” classifier is the CRM114 OSB classifier which was introduced in 2003 and is substantially unchanged since 2004. We use the OSB classifier as a well-known benchmark to make an inter-year comparison of the difficulty of each new test set.

The basic concepts of Markov OSB classification is to break the text down into words (where a word is defined as a set of printing characters separated by whitespace characters) then to assemble short pseudophrases, by skipping zero to three words, from consecutive wordsets. For example, the text

```
“Gentlemen!  You can't fight in here!  This is the War Room!”
```

will break down into the feature pseudophrases:

```
Gentlemen! You
Gentlemen! <skip> can't
Gentlemen! <skip> <skip> fight
Gentlemen! <skip> <skip> <skip> in
You can't
You <skip> fight
You <skip> <skip> in
You <skip> <skip> <skip> here!
can't fight
...
```

and so on (the <skip> tokens are significant, as is case and word order; the features “You can't” and “can't You” are distinct. There is no stop word removal, and no preloaded dictionary; everything is learned from zero.

Once these features are created, they are hashed and a lookup is done; from the ratio of times each feature is seen in the spam and nonspam corpora, a probability is calculated (heavily biased towards 0.5 especially when the number of seen occurrences is small) and the probabilities are combined with a normalization rule such as Bayes law. Overall, the mathematical structure formed from the known feature pseudophrases is a Markov random field and the text stream steps across that manifold; the combining rule measures how well the given text is modeled by the field.

Classifier Details: SVM : The SVM classifier is a C-SVM based on the well-known work of Boser et al 1992 and Cortes and Vapnik 1995). We use an SMO type decomposition [Fan, Chen, and Lin 2005] to select the working set.

In prior work by other researchers, the typical “feature” in the SVM was an individual word – typically this meant that the most common words only were used in the SVM. Instead of this, we performed the SVM using entire documents as single features; thus the output hyperplane of the CRM114 SVM is a

weighting of total documents rather than single words. [1]

In more detail, the CRM114 SVM classifier performs the OSB feature creation as seen above, and then keeps the entire document's hash set as a single very large feature; this is the same behavior and code as the CRM114 “Hyperspace” K-nearest-neighbor classifier. To calculate the distance between any two such documents, we simply count the number of shared pseudophrase hashes, which is the number of shared pseudophrases as described above under OSB. This is the “dot” function we use in the C-SVM.

Solving for the optimal SVM weighting of these features is done on a pairwise basis; pairs of points are selected and the hyperplane weights are adjusted to maximise the error margin between them. The process repeats until convergence. The result is a set of a few documents in each class whose dot-product weights are nonzero; these define the separation hyperplane.

Unlike the prior “Hyperspace” KNN code that is completely nonlinear, the current SVM code produces a linear classifier. Because the SVM calculation determines which support vectors (that is, documents) are significant and which are not significant, the SVM code is capable of forgetting documents that are insignificant and likely to remain insignificant.

In more detail, we implemented one type of SVMs called C-Support Vector Classification (C-SVC). The dual formulation of C-SVC is to find

$$\min \quad 0.5 (\quad ^{T} Q \quad) - e^{T}$$

$$\text{subject to} \quad y^{T} = 0$$

$$y_i = +1 \text{ or } -1$$

$$0 <= \quad _i <= C, \; i=1,...,\text{sizeof(corpus)}.$$

where "e" is the vector of all ones, Q is the sizeof(corpus) by sizeof(corpus) matrix containing the calculated distances between any two documents (that is, $Q_{ij} = y_i * y_j * \text{kernel}(x_i, x_j)$ which may be intractably large and so we only calculate part of it at any one time (x_i is the feature vector of document i). The decision function is

$$\text{sgn} (\text{sum}(y_i * \quad i * \text{kernel}(x_i, x_j) + b)$$

In order to solve the above quadratic optimization problem, we used SMO-type decomposition method proposed by Fan et al. [2] in which each iteration chooses two elements in the vector based on second order information and only updates these two elements instead of the whole vector. But for some email corpuses, this method will choose the same element in several times and then converge very slowly. So in our implementation, for every element in we set a upper bound to stop it from

being chosen for update too many times.

In our experiments, we tried different kernel functions on TREC 2006 : linear kernels, polynomial kernels, and a radial basis function (RBF). Our testing shows that linear kernel performs better than the other two.

Compared with other classification methods, one advantage of SVM is that the hyperplane is only determined by a small number of support vectors (only a few elements in vector aren't zero). In another words, for a new email we can quickly determine whether it is spam or not by calculating the decision function. The second advantage of SVM is that it can handle unbalanced datasets in which different classes have very different training sizes because the algorithm does not directly try to minimize the error rate within classes but try to separate data in a higher dimension space.

Classifier Details: Bit-Entropy: The bit-entropy classifier is a bit-at-a-time compression scheme. This classifier is rather unique in that not only doesn't it have any concept of a text string; it doesn't even have any concept of a byte. During training, the serialized bits of incoming known text are used to generate a transition lattice that is later used to optimally compress the unknown text. Because the compressed text is not actually used, this classifier only calculates the absolute entropy of the result; the transition lattice that produces the best compression is the one that represents the unknown text most closely and hence was produced by members of the unknown text's class.

To produce a compression lattice rather than a simple directed tree, we provide the ability to crosslink during the construction of the transition lattice. Specifically, if we are about to add a node, but a node with a similar prior transition history already exists, we do not allocate a new node, but instead route the current transition path to the old node. This allows our compression lattice to have arbitrary loops and cycles as required to describe the text documents.

As an improvement to this, we also check the tentative new node's future paths; our current system looks a minimum at 20 bits of past history and three bits of future (hence one could consider the compression coder is "three bits precognitive").

One advantage of this method of generating the compression lattice "on the fly" is that the amount of memory needed is quite small compared to other methods; we typically run with only one million bit nodes total and find that more than adequate for the TREC public corpus.

Classifier Details: Substring Compression: The fourth CRM114 classifier tested in TREC2007 is the Fast Substring Compression Match (FSCM) classifier. This classifier is based on the Lempel-Ziv compression algorithm [3] commonly known as LZ-77 . In the CRM114 implementation, the prior-text window is greatly expanded (with a default of one megabyte for the TREC runs), a minimum compressible string length of three characters in succession, and some additional data structures are

used to accellerate the processing, but in general the algorithm is unchanged.

For scoring, the document is deflated and for each substring match that would cause LZ-77 to record a repeated substring, the number of bytes that would be compressed out is raised to the 1.5th power and added to the score (the 1.5 value is empirically determined). Futher, we do not do any Huffman coding on the unrepeated substrings, as that would reflect only individual character frequencies rather than string-to-string similarities in the texts. Greater overall score implies greater overall similarity between the known and unknown texts.

Results:

The main TREC results can be summarized in the following table giving overall error rate and 1-ROCA% value and confidence inter(1-ROCA% is the area not under the ROC curve expressed as a percentage.:

Test Set	Magnitude of test set	CRM run 1 (SVM)	CRM run 2 (OSB)	CRM run 3 (bit entropy)	CRM run 3 (string compression)
MrX Full	161975	**202 (99.875%) ROCA: 0.0543 (0.0297 - 0.0994)**	287 (99.822%) ROCA: 0.1296 (0.0750 - 0.2238)	548 (99.661%) ROCA: 0.9476 (0.7943 – 1.130)	956 (99.409%) ROCA: 0.1145 (0.0807-0.1626)
MrX Delayed Feedback	161975	**550 (99.659%) ROCA: 0.2079 (0.1515 - 0.2852)**	1208 (99.254%) ROCA: 0.3811 (0.3006 - 0.4829)	2395 (98.52%) ROCA: 2.516 (2.2869 - 2.7693)	4761 (97.06%) ROCA: 0.7589) (0.7106 - 0.8105)
Public Full	75397	**93 (99.876%) ROCA: 0.0142 (0.0067 - 0.0302)**	158 (99,790%) ROCA: 0.1289 (0.1032 - 0.1611)	308 (99.591%) ROCA: 0.2364 (0.1951 - 0.2864)	240 (99.681%) ROCA: 0.0457 (0.0306 - 0.0683)
Public Delayed Feedback	75397	**242 (99.679%) ROCA: 0.0229 (0.0147 - 0.0356)**	875 (98.839%) ROCA: 0.3186 (0.2607 - 0.3893)	1161 (97.863) ROCA: 0.8844 (0.8049 - 0.9716)	344 (99.543%) ROCA: 0.3354 (0.2759 - 0.4076)
Public Partial Feedback	75397	**853 (98.868%) ROCA: 0.0425 (0.0147 - 0.0356)**	2357 (96.873%) ROCA: 0.3882 (0.2607 - 0.3893)	2979 (96.05%) ROCA: 0.6743 (0.8049 - 0.9716)	1943 (97.42%) ROCA: 0.1866 (0.2759 - 0.4076)

To our mild surprise, we found that the SVM did the best overall in all testing regimes (bold in the table above), in two cases, well within the error bars to be “best overall” for TREC 2007.

Discussion:

We were pleasantly surprised at the good showing by the SVM classifier; by any measure we see it performs better than any other CRM114 classifier on the TREC corpora. (1-ROCAs of 0.05 and 0.01 versus OSB at 0.12 and 0.13, and 1/3 fewer total errors. Interestingly, the SVM is working against the very same data representation and format as the weak-performing Hyperspace KNN filter of TREC 2006, which did only a ROCA of 0.2217 (0.1682 to 0.2923 at TREC 2006.

Our canary-in-the-coal-mine standard, the 2003 vintage OSB classifier on the new Mr X version 3 (2007) corpus confirmed again that spammers are not getting smarter; OSB got slightly better 1-ROCA of 0.128 (0.077 to 0.22) versus 0.1498 (0.105 to 0.231) for MrX Version 2 (2006) and 0.177 (0.128 to 0.246) for Mr X version 1 (2005).

Bit Entropy was a losing proposition at TREC 2007. The addition of “precognition” did not seem to improve performance in either ROC terms or in absolute error rate over the non-precognitive version in TREC 2006 (performance actually decreased from the TREC 2006 numbers, but not by a statistically significant amount). We will continue to consider algorithmic improvements to bit entropy in the future, and for problem corpora that do not contain blank-delimited words.

Finally, the FSCM (Fast Substring Compression Matching, via LZ-77) did surprisingly well. In the Public Full corpus, it beat OSB by a strongly statistically significant amount (the error bars do not overlap); in the private Mr. X corpus, it did better than OSB but not by a statistically significant amount.

The results also as some surprises, such as some inversions between raw error rates and 1-ROCA (the like-colored pairs above. In particular, note the results of the MrX Full corpus for bit entropy and OSB versus substring compression: if one measures by 1-ROCA, compression is eight times better, but measuring by actual error count shows that it's almost twice worse; their confidence intervals don't even overlap.

References:

[1] Richard O.Duda, Peter E.Hart,. David G.Stork (2004). *Pattern Classification*, pages 259 – 265

[2] R.-E. Fan, P.-H. Chen, and C.-J. Lin (2005). *Working set selection using second order information for training SVM*. Journal of Machine Learning Research, 6:1889-1918.

[3] Jacob Ziv and Abraham Lempel; A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory, 23(3), pp-337-343, May 1977