SpamAssassinChallenge

The SpamAssassin Challenge

(THIS IS A DRAFT; see bug 5376 for discussion)

The Netflix Prize is a machine-learning challenge from Netflix which 'seeks to substantially improve the accuracy of predictions about how much someone is going to love a movie based on their movie preferences.'

We in SpamAssassin have similar problems; maybe we can solve them in a similar way. We have:

- a publishable large set of test data
- some basic rules as to how the test data is interpreted
- a small set of output values as a result
- which we can quickly measure to estimate how "good" the output is.

Unfortunately we won't have a prize. Being able to say "our code is used to generate SpamAssassin's scores" makes for good bragging rights, though, I hope 😉

Input: the test data: mass-check logs

We will take the SpamAssassin 3.2.0 mass-check logs, and split them into test and training sets; 90% for training, 10% for testing, is traditional. Any cleanups that we had to do during bug 5270 are re-applied.

The test set is saved, and not published.

The training set is published.

Input: the test data: rules, starting scores, and mutability

We can provide "tmp/rules_*.pl" (generated by "build/parse-rules-for-masses"). These are perl data dumps from Data::Dumper, listing every SpamAssassin rule, it's starting score, a flag indicating if the rule is mutable or not, and other metadata about it.

Mutability

Mutability of rule scores is a key factor. Some of the rules in the SpamAssassin ruleset have immutable scores, typically because:

- they frequently appear in both ham and spam (therefore should not be given significant scores)
- or we have chosen to "lock" their scores to specific values, to reduce user confusion (like the Bayes rules)
- or to ensure that if the rule fires, it will always have a significant value, even though it has never fired yet (the "we dare you" rules)
- or we reckon that the rule's behaviour is too dependent on user configuration for the score to be reliably estimated ("tflags userconf" rules)

This mutability is defined by us up-front, by selecting where the rule's score appears in "rules/50_scores.cf" (there are "mutable sections" and "immutable sections" of the file).

In addition to this, some rules are forced to be immutable by the code (in "masses/score-ranges-from-freqs"):

- rules that require user configuration to work ("tflags userconf")
- network rules (with "tflags net") in score sets 0 and 2
- trained rules (with "tflags learn") in score sets 1 and 3

Some scores are always forced to be 0 (in "masses/score-ranges-from-freqs"). These are:

- network rules (with "tflags net") in score sets 0 and 2
- trained rules (with "tflags learn") in score sets 1 and 3

(Rules with scores of 0 are effectively ignored for that score set, and are not run at all in the scanner, so this is an optimization. If you don't know what a score set is, see MassesOverview.)

In addition, rules that fired on less than 0.01% of messages overall, are forced to 0. This is because we cannot reliably estimate what score they *should* have, due to a lack of data; and also because it's judged that they won't make a significant difference to results either way. (Typically if we've needed to ensure such a rule was active, we'd make it immutable and assign a score ourselves.)

During SpamAssassin 3.2.0 rescoring, we had 590 mutable rules, and 70 immutable ones.

TODO: we will need to fix the tmp/rules*.pl file to reflect the limitations imposed in this section, or generate another file that includes these changes.

Score Ranges

Currently, we don't allow a rescoring algorithm to simply generate any score for a mutable rule at all. Instead we have some guidelines:

Polarity: scores for "tflags nice" rules (rules that detect nonspam) should be below 0, and scores for rules that hit spam should be above 0. This is important, since if a spam-targeting rule winds up getting a negative score, spammers will quickly learn to exploit this to give themselves negative points and get their mails marked as nonspam.

No single hit: scores shouldn't be above 5.0 points; we don't like to have rules that immediately mark a mail as spam.

Magnitude: we try to keep the maximum score for a rule proportional to the Bayesian P(spam) probability of the rule. In other words, a rule that hits all spam and no ham gets a high score, and a rule that fires on ham 10% of the time (P(spam) = 0.90) would get a lower score. Similarly, a "tflags nice" rule that hits all ham and no spam would get a large negative score, whereas a "tflags nice" rule that hit spam 10% of the time would get a less negative score. (Note that this is not necessarily the score the rule will get; it's just the maximum *possible* score that the algorithm is allowed to assign for the rule.)

These are the current limitations of our rescoring algorithm; they're not hard and fast rules, since there are almost definitely better ways to do them. (It's hard to argue with "Polarity" in particular, though.)

Output: the scores

The output should be a file in the following format:

```
score RULE_NAME    1.495
score RULE_2_NAME  3.101
...
```

Listing the rule name, and score, one per line, for each mutable rule. We can then use the "masses/rewrite-cf-with-new-scores" script to insert those scores into our own scores files, and test FP% / FN% rates with our own test set of logs.

Runtime Limits

The code needs to be "fire and forget" automated; hand-tweaking of settings must not be necessary. It must be possible to just give it the "rules_N.pl", let it gronk, and get the scores output.

Evaluation Criteria

TODO, still talking about this

Licensing

The code produced would need to be usable without any sort of patent license, and available under the Apache Software License 2.0 (ie. suitable for inclusion in the Apache SpamAssassin source tree).