
Attacking SpamBayes: Compromising a Statistical Spam Filter

Marco Barreno Fuching Jack Chi Anthony D. Joseph Blaine Nelson
Benjamin I. P. Rubinstein Udam Saini Charles Sutton J. D. Tygar Kai Xia

Computer Science Division, UC Berkeley

1 Problem and Methodology

We explore the resilience of machine learning applications in adversarial environments by developing attacks against a real-world learning application, using influence over the learner’s training data to corrupt the learned model. There is a growing body of work in this area, but due to space constraints we will limit ourselves to describing our own contributions here. The application we investigate is SpamBayes [2], a popular¹ open-source spam filter based on a naive Bayes learning algorithm over word probabilities. SpamBayes trains from labeled data then scores each incoming email, comparing against two thresholds to label it as *spam*, *non-spam*, or *unsure*. We assume that the attacker controls spam training emails, and for some attacks also some non-spam training emails.

The core insight underlying our first set of attacks is that each incoming non-spam email will (with high probability) contain words that occur rarely or not at all in the training set. If the attacker can guess these words and include them in attack emails that are trained as spam, this will cause some non-spam emails to be classified as spam; enough such misclassifications create a *denial of service* (*DoS*) attack by making SpamBayes unusable and causing the user to disable it. We call these attacks *dictionary attacks* after the naive version that uses the entire English dictionary in the attack.

We are exploring several variants of the dictionary attack based on different means of choosing the attack words:

Naive dictionary attack. We include the entire dictionary of words. This ensures coverage of rare words in incoming emails (subject to the completeness of the dictionary).

Distribution attack. Given a distribution over all words and the size of the training set, we calculate words most likely to occur soon after training while being absent from the training set.

Targeted attack. This attack targets a particular type of email about which some information is known; we include words likely to be in the target email.

The second type of attack is aimed at causing spam emails to pass through the filter and get to the user’s inbox. We have two variants:

Pseudospam attack. We include messages in the non-spam training set that come from a spammer but do not appear to be spam (e.g. random text). Similar emails with advertising may not be filtered, as the learned non-spam weights outweigh the advertisement.

Dilution attack. We spread the words of a target spam across many non-spam training emails, placing a small number of words in each. Each word in the target spam gains some non-spam weight.

¹SpamBayes’s Sourceforge project page reports over 670,000 downloads of the Windows executable.

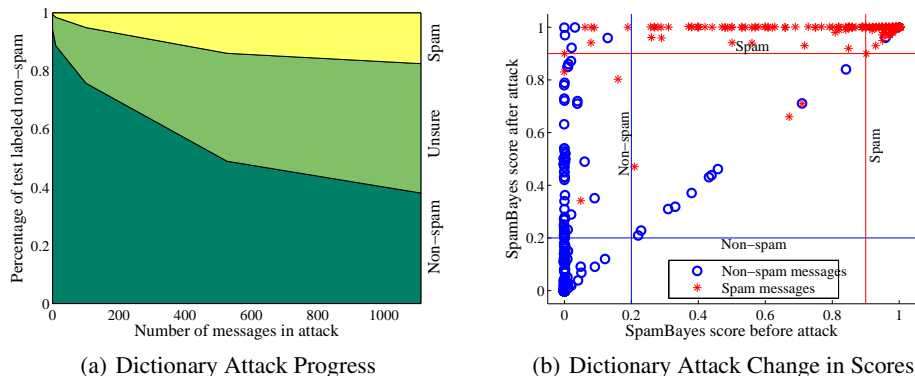


Figure 1: Results from naive dictionary attacks on a 10,000-message corpus. Figure 1(a) shows changes in classification as an attack progresses. The yellow (top), light-green (middle), and dark-green (bottom) areas depict the fraction of non-spam emails classified as spam, unsure, and non-spam, respectively, as the attack progresses. Figure 1(b) shows message scores before and after an attack of 50 emails. Lines mark the SpamBayes spam/unsure and non-spam/unsure thresholds.

2 Preliminary Results

We injected a naive dictionary attack into an email dataset sampled from the TREC’05 corpus [1], which is based on the Enron email dataset and contains 92,189 emails (52,790 spam and 39,399 non-spam). We used training corpora of 10,000 emails with 60–90% spam. We injected dictionary attack emails into our training set accounting for 0.1% to 10% of the training emails. Note that although the attack emails make up a small percentage of the *number of messages* in a poisoned inbox, they make up a large percentage of the *number of tokens*. For example, 200 attack emails are just under 2% of the emails in an experiment but make up around 80% of the tokens.

Empirically, the dictionary attack successfully caused non-spam emails to be misclassified. As shown in Figure 1(a), SpamBayes misclassified more non-spam as our attack progressed. We further examined the change in scores for our held-out test emails before and after the attack. As shown in Figure 1(b), the spam score of many emails increased (often dramatically) because of the attack, causing many non-spams to fall into the unsure and spam categories.

The success of our attack relies on its coverage of words in the tail of the user’s word distribution. Using an entire English dictionary gives high coverage, making the naive dictionary attack statistically effective although unwieldy in practice. Other attacks described above explore using informative distributions over attack words to reduce the number of words required without decreasing the effectiveness.

3 Future Work and Open Problems

We continue to develop and refine the attacks presented here. These attacks demonstrate empirically that practical learning systems are vulnerable to malicious attack. Furthermore, we are developing defenses against these attacks; such countermeasures are vital to the use of learning in real systems. Promising directions include pre-processing training data to filter out attacks, disregarding or down-weighting points with undue influence over training, and applying methods from robust statistics to prevent outliers or incorrect model assumptions from excessively skewing the learner.

References

- [1] G. Cormack and T. Lynam. Spam corpus creation for TREC. In *Conference on Email and Anti-Spam (CEAS)*, 2005.
- [2] T. A. Meyer and B. Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *First Conference on Email and Anti-Spam (CEAS)*, 2004.