A Linguistics-Based Attack on Personalised Statistical E-mail Classifiers

Henry Stern
Justin Mason
Michael Shepherd

Technical Report CS-2004-06

March 25, 2004

Faculty of Computer Science
6050 University Ave., Halifax, Nova Scotia, B3H 1W5, Canada

A Linguistics-Based Attack on Personalised Statistical E-mail Classifiers

Henry Stern
Faculty of Computer Science
Dalhousie University
6050 University Avenue
Halifax, NS, Canada B3H 1W5
stern@cs.dal.ca

Justin Mason
Network Associates Inc.
3965 Freedom Circle
Santa Clara, CA, USA 95054
Justin_Mason@NAI.com

Michael Shepherd
Faculty of Computer Science
Dalhousie University
6050 University Avenue
Halifax, NS, Canada B3H 1W5
shepherd@cs.dal.ca

March 25, 2004

Abstract

We present a potential vulnerability of personalised anti-spam filters where an attacker sends carefully constructed e-mail messages with the goal of negatively affecting classifier accuracy. Words from the core of the English language are randomly "injected" into spam e-mails for the express purpose of manipulating the probability tables of a naïve Bayesian classifier. This attack method is shown to be successful in reducing classifier accuracy within a laboratory environment. Barriers to a real-world implementation and potential countermeasures are discussed.

1 Introduction

On the surface, unsolicited commercial e-mail (spam) classification is a very easy text classification problem. Countless papers on the subject have been written, evaluating the performance of almost every available text classification algorithm. However, very few of these papers address the adversarial

nature of the problem, i.e., people sending unsolicited commercial e-mail (spam) do not want it to be classified as such.

While the concept was initially proposed in 1998 by Sahami, Dumais, Heckerman and Horvitz [1] and Pantel and Lin [2], an essay published in 2002 by Paul Graham [3] sparked public interest in the idea of training personalised text classifiers to identify one's spam e-mail. Two years later, most of the widely-used spam detection products incorporate some kind of personalised text classifier with the majority using incrementally-trained naïve Bayes classifiers. These products include SpamAssassin[1], McAfee SpamKiller[2], Mozilla Thunderbird[3], Apple Mail[4], POPFile[5] and SpamBayes[6].

The widespread adoption of personalised text classifiers has prompted the spam senders to develop a technique dubbed "Bayes poison" which involves adding words and phrases to the body of an e-mail. Several variants of this attack are discussed in Sophos Inc.'s *Field guide to spam* [4]. This attack exploits the fact that most text classification algorithms treat a message as a "bag of words" where a human does not. The intent of this attack is to cause the user's classifier to falsely accept a single spam e-mail.

Graham-Cumming [5] has proposed an attack pattern where an attacker sends thousands of e-mails to an individual user containing "web bugs" to track which messages are able to penetrate that individual's personalised filter. Considering that most spam senders have mailing lists consisting of millions of users, sending and tracking trillions of e-mail messages to individual users is highly impractical.

We propose a more general attack pattern with the goal of reducing the quality of service provided by the personalised e-mail filters. Instead of targeting individual users with thousands of e-mail messages, spam senders add random words from a carefully selected vocabulary to their messages.

Zipf's Law [6] suggests that the frequency of usage of a word in a language is inversely proportional to its ranking. This implies that there is a subset of every language that is used extensively by every user. Since they convey no useful information, the most common words are usually ignored as noise words. However, there is still a significant overlap in the vocabularies of the users of a language that conveys information for classification.

[1] SpamAssassin: `http://www.spamassassin.org/`
[2] McAfee SpamKiller: `http://www.nai.com/`
[3] Mozilla Thunderbird: `http://www.mozilla.org/products/thunderbird/`
[4] Apple Mail: `http://www.apple.com/macosx/features/mail/`
[5] POPFile: `http://popfile.sourceforge.net/`
[6] SpamBayes: `http://spambayes.sourceforge.net/`

The vocabulary commonly used by spam e-mail is fairly limited. Prominent words include "sex," "mortgage" and the names of various pharmaceuticals and herbal products. Personalised text classifiers learn which of these particular words are not in the user's active vocabulary and use this to their advantage for classification.

If the most voluminous senders of spam were to purposely add large numbers of words from the common vocabulary in the body of their messages and end-users were to train their classifiers on those messages, those commonly-used words would become indicative of spam rather than being indicators of legitimate e-mails or simply benign in the classification process. A typical use case would be where a spam sender chooses a set of random words and then works them into the actual content of their e-mail.

This attack is general for all users of a particular language. We implement such an attack on a public benchmark corpus and study the long-term effects on classifier effectiveness.

2 Poisoning Classifiers

To determine the effectiveness of our proposed attack, we perform a set of experiments comparing two configurations of a naïve Bayes text classifier on a public benchmark corpus where "poison," i.e., terms from the core of the English language, is injected.

We use a naïve Bayesian [7] classifier with Laplace smoothing [8]. For one configuration we do not use a feature selection algorithm and for the other, we use an entropy-based feature selection algorithm. We use an event-based model where a feature is present only when a corresponding word occurs in the document. While it is well studied in the literature that feature selection is important for text classification with naïve Bayes, it reflects the real-world behaviour of the majority of the current spam classifiers. The entropy-based feature selection algorithm removes features with high entropy and preserves those with high information. This method is used because it is efficient in an incrementally trained environment and is independent of message length.

We evaluate the attack pattern using the *bare* variant of the ling-spam benchmark corpus created by Androutsopoulos et. al. [9] Ling-spam consists of 2412 legitimate e-mail messages and 481 spam e-mail messages, divided into 10 partitions for cross validation. There are four variants of the ling-spam corpus: *bare*, *lemm*, *stop*, and *lemm-stop*. These variants were used by Androutsopoulos et. al. to investigate the effects of lemmatisation and stop-words on classifier accuracy.

Configuration	Spam Precision	Spam Recall
No poison, all features	0.8470	0.9959
Poison, all features	0.6714	1.0000
No poision, feature pruning	0.9449	0.9959
Poison, feature pruning	0.8447	1.0000

Table 1: Performance statistics for the various classifier configurations.

We use the *1000 Most Common Words in English* [10] obtained from *English as a Second Language*, published online by About.com to inject "poison" into the spam e-mails of the training and test sets of ling-spam. We randomly select two words, with replacement, from the list, and add them to the subject of the e-mail, and 100 words, with replacement, to add to the body of the spam e-mail. The words are sampled uniformly, rather than proportionally to their ranking. This is done to further upset the distribution of words in the corpus. Since we are evaluating performance using a classifier that uses an event-based model, the specific locations of these words within the text is not considered.

Of the 1000 terms contained in the About.com ESL corpus, 982 of them occur in the messages in the ling-spam corpus and account for 43% of all term occurrences. Given that the two corpora were developed independently, this lends credibility to the assumption that the majority of English speakers, in general, tend to focus their linguistic usage on these 1000 terms.

For each configuration of the classifier and data set, we perform a 10-fold cross validation using the folds and report the precision and recall within the spam class.

3 Results

The results of the experiments are presented in Table 1. Spam precision is defined as the proportion of messages that are classified as spam that are, in fact, spam. Spam recall is the proportion of spam messages that are correctly classified as spam. In all of the experiments, the classifier was able to identify almost all of the spam e-mails but mistakenly classified a large number of legitimate e-mails as spam. As expected, feature pruning offered a statistically significant increase in precision for both the unmodified ling-spam and poisoned ling-spam corpora.

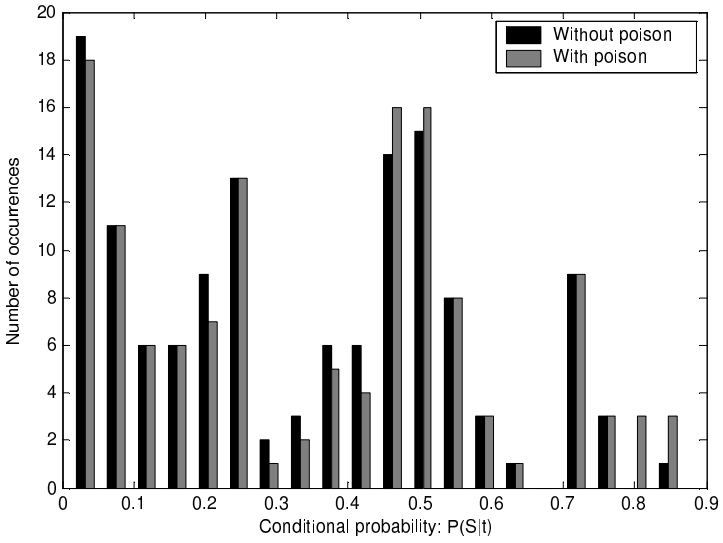

Figure 1: Histogram of conditional probabilities for terms occurring in a legitimate message.

The precision of the classifier on the poisoned corpus is statistically significantly lower than that of the un-poisoned corpus in each case. This occurs because the poisoning process has been successful in manipulating the probability tables to bias towards the spam class. Figure 1 shows the histogram of the conditional spam probabilities of the terms in a single legitimate e-mail message. It illustrates how the poisoning process manipulates the estimated conditional probabilities. The estimated conditional probabilities for the injected terms are increased, causing the classifier to bias towards the spam class. This phenomenon occurs for every message in both classes of the corpus.

Another observation is that the general shape of the frequency distribution has been maintained. The probability estimates for the un-injected terms remain constant and the probability estimates are dramatically increased for the injected terms. This is important because many of the terms that should be considered good indicators of whether a message is spam are now in-discriminable from the terms that make up the majority of the language usage.

A consequence of the poisoning process is that spam e-mail messages actually have a lower probability of being accepted as legitimate e-mail and

legitimate e-mail has a higher probability of being mistakenly rejected as spam e-mail. The classifier will consistently bias its classifications towards the spam class for every message containing the injected terms.

$P(t|S)$ and $P(t|L)$ are defined as the proportions of spam and legitimate e-mail messages containing the term, t. $P(S)$ and $P(L)$ are defined as the proportions of spam and legitimate e-mail messages in the corpus. $P(t)$ is defined as the probability that a term, t, occurs in a message. The conditional probability that a message is spam given that it contains a single term, t, is defined in equation 1. The conditional probability for a message being legitimate is similarly defined.

$$P(S|t) = \frac{P(t|S)P(S)}{P(t|S)P(S) + P(t|L)P(L)} \tag{1}$$

By artificially increasing the frequencies of common words in the spam messages of the training set, $P_{poisoned}(t|S)$ increases while $P(t|L)$ remains constant and $P(S|t) \leq P_{poisoned}(S|t)$. The conditional probability for the entire message is computed using equation 2.

$$P(S|T) = \frac{P(S) \prod_{t \in T} P(S|t)}{P(S) \prod_{t \in T} P(S|t) + P(L) \prod_{t \in T} P(L|t)} \tag{2}$$

As $P(S|t) \leq P_{poisoned}(S|t)$ and $P(L|t) \geq P_{poisoned}(L|t)$, for every message classified $P(S|T) \leq P_{poisoned}(S|T)$. Therefore, the classifier will bias every message containing any term that has been injected towards the class of spam e-mails.

4 Conclusions

We have shown that injecting common words into spam e-mail can degrade classifier performance in a controlled laboratory environment. Implementing such an attack in the real world would be significantly more difficult.

From a social standpoint, a real-world implementation of this attack would require the cooperation of all of the most active senders of spam e-mail. The implementation of this attack is equivalent to the non-zero sum game, "Prisoner's Dilemma." [11] If any of the major spam senders were to "fink," their e-mails would have a greater probability of reaching their intended audience than that of their competitors and would greatly reduce the effectiveness of the attack by reducing the amount of bias induced in the classifiers.

From a technical standpoint, an attack such as this would take a very long time to produce any measurable effect on the accuracy of the deployed spam filters. These filters are trained incrementally and thus will have already computed stable conditional probability estimates. The effectiveness of this attack on existing filters would be inversely proportional to the amount of un-polluted data that the filter has been trained on. However, new filters that had never received un-polluted training data would be affected by this attack.

An obvious countermeasure to this problem is to treat all of the most common words in the language as noise words and rely on lower-frequency terms such as proper nouns for classification. Classifiers will be more susceptible to sampling noise. In some extreme cases, this will cause the entire body of an e-mail message to be ignored requiring the decision to be made using only non-textual information from the message headers.

A more viable solution would be to combine the information gain of a term with its "trustworthiness." In this case, the trustworthiness of a term is defined as the probability that a stranger can guess whether that term is part of an individual user's active vocabulary. Without intimate knowledge of an individual, familiar proper names and interests are virtually impossible to guess by a stranger. To use this information in a statistical classifier, the estimated conditional probabilities for the least trustworthy terms are smoothed towards 0.5 and the conditional probabilities for the most trustworthy terms are allowed to remain at their original estimates.

In conclusion, injecting terms into an e-mail message actually causes fewer spam messages to pass through personalised e-mail filters undetected while causing significantly more legitimate messages to be rejected. Damage from a large-scale attack can be minimised through consideration of the trustworthiness of an estimated probability. While this type of attack should not be considered an immediate danger, the concept of trustworthiness of terms should be further investigated to preserve the future integrity of personalised e-mail filters.

References

[1] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *AAAI Workshop on Learning for Text Categorization*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[2] Patrick Pantel and Dekang Lin. Spamcop: A spam classification & organization program. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[3] Paul Graham. A plan for spam. `http://www.paulgraham.com/spam.html`, August 2002. Last accessed March 2, 2004.

[4] Sophos Inc. Field guide to spam. `http://www.sophos.com/spaminfo/explained/fieldguide.html`. Continuously updated. Last accessed March 2, 2004.

[5] John Graham-Cumming. Fooling and poisoning adaptive spam filters. Technical report, Sophos Inc, November 2003.

[6] George Kingsley Zipf. *Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology*. Addison-Wesley, 1949.

[7] Tom M. Mitchell. *Machine Learning*. McGraw Hill, New York, US, 1996.

[8] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, 2000.

[9] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, George Paliouras, and Constantine D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proceedings of the workshop on Machine Learning in the New Information Age*, 2000.

[10] Kenneth Beare. Vocabulary workshop: 1000 most common words in english. `http://esl.about.com/library/vocabulary/bl1000_list1.htm`. Last accessed March 2, 2004.

[11] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, 1985.