

# Social Networking Spam: Will You Be My Friend?

Jonathan Kupferman  
Department of Computer Science  
University of California, Santa Barbara  
Santa Barbara, CA 93106  
jkupferman@cs.ucsb.edu

Jeffery Silverman  
Department of Computer Science  
University of California, Santa Barbara  
Santa Barbara, CA 93106  
jdsilverman@cs.ucsb.edu

## ABSTRACT

We begin by discussing the recent trend towards spam on Social Networking web sites. We then study the performance of three types of classifiers in order to detect spam on these sites. While Naïve Bayes has been a mainstay in text-categorization and spam filtering, we compare its performance to a boosting algorithm and a decision-tree algorithm. While boosting algorithms show improvements over Naïve Bayes, decision-trees provide more accurate spam detection than both. Decision-trees are able to achieve strong results using a very small number of features.

## Categories and Subject Descriptors

K.4.m [Computers and Society]: Miscellaneous; H.4.m [Information Systems]: Miscellaneous

## General Terms

Measurement, Experimentation, Algorithms

## Keywords

Social Network, Spam

## 1. INTRODUCTION

Large-scale email spam has become somewhat commonplace in the Internet's short history. Yet, improvements in spam detection technology have significantly reduced the rate at which spam is able to reach a users inbox (and not be relegated to the spam folder). Given that this is the case, spammers have been actively searching for new and more effective ways to deliver their messages. By far the most important criteria for selecting a medium to spam is first and foremost the ability to reach millions of people, since the spam market is noted to have an extremely low return rate ( $>0.00001\%$ ) per message sent[12]. If a single message can be seen by more than one user (e.g. forum post), this serves as an added bonus for the spammer. These criteria have lead spammers to Social Networking websites, many of which have over 100 million subscribers and frequent visits from users[5].

## 2. RELATED WORK

There has been a significant amount of research in the area of email spam detection dating as far back as 1998 with the work of Sahami, Dumais, Heckerman, and Horvitz on "A Bayesian Approach to Filtering Junk E-mail." [24] Then in 2002 Paul Graham's "A Plan for Spam" [8] marked the beginning of heavy research into the use of Bayesian techniques for email spam detection. The early works proposed the use of Naive Bayes classifiers with the multi-variate Bernoulli with Boolean attributes [16, 22, 24]. The work of Pantel and Lin also use Naive

Bayes, but with multinomial attributes [19]. There have been many other improvements proposed for Naive Bayes including Gaussian methods, these methods will not be discussed here as they have already been covered in detail elsewhere [8, 9, 16, 17, 22, 24].

As spam has spread beyond the email inbox new areas of spam detection have arose, each with different constraints. Traditional spam detection mechanisms have proven less effective on Instant Messaging and SMS spam both of which are "rife with abbreviations and idioms." [3] It has been noted that in order to improve results on shorter messages it is recommended that the feature set should be expanded. [3, 4, 11]

## 3. EXPERIMENTS

### 3.1 Background

There are some significant differences between communication via email and Social Networks which leads to a large changes in the way spamming is done. One of the most noted differences between the two is that with email, if ones email address is known, a message can be sent to them. Most Social Networks require that in order to send someone a message one must be their "friend" on the given site, and becoming a friend generally requires that the user accept or approve a "friend request". Furthermore, friend status can be revoked, thus disallowing further messages to be sent between the former friends. This screening mechanism presents a significant barrier of entry for spammers. It is likely that many users will decline or not respond to a spammers friend requests, and the ones who do can easily remove the spammer from their friend list. Many spammers instead exploit the fact that some groups on Social Networks, especially larger ones, do not require approval to join. As a result spammers are able to join a group and immediately post messages on the groups "Message Board." This technique allows the spammer to post messages which are viewable by hundreds or even thousands of group members, without having to be accepted as a friend of each one.

### 3.2 Dataset

In order to study spam on Social Networking sites we were able to gather message posts from Facebook, one of the world's largest Social Networking websites with over 130 million users [5]. The posts were collected from both "The Wall" and "Discussion Board" of all public groups and applications with at least 5000 members. The corpus contains over 12 million messages which were posted between July of 2006 and October 2008, when the messages were collected. The training set was composed of over 7920 messages randomly selected messages from the corpus. Of those messages, 1384 (17%) were considered spam while 6478 (82%) were considered "ham" (i.e. not spam).

### 3.3 Judging Criterion

In order to judge an algorithm it is run on over set of data which has already been categorized called the *training set*. Before it is run, some portion of the training set is removed such that the given algorithm can attempt to categorize those messages. The results predicted by the classifier are then compared to the actual results and the cases in which they differentiate are noted. The percentage of classifications the algorithm makes correctly versus the total number of classifications provides a general overview of how a given algorithm performed. It does not however take into account the differences in the cases where it was incorrect; these differences are categorized using the *precision* and *recall* metrics[2, 17]. High precision indicates that there are few cases where messages are incorrectly classified as spam (i.e. false positives). High recall, on the other hand, indicates that there are few cases where messages were incorrectly classified as ham (i.e. false negatives). Given that it is significantly worse to for a ham message to get classified as spam, than vice-versa, precision is considered more important than recall.[8]

### 3.4 Classifiers

We use Weka[28], a data-mining library, to classify the data since it contains a large set of data classification and mining algorithms. Traditionally, email spam detection has relied on Bayesian techniques, namely Naive Bayes, which has proven to be simple and effective. Many other classification techniques have been used which have shown improvements over Naive Bayes. In order to experiment with different types of classifiers, we chose two additional classifiers which are representative of two other types of classification algorithms.

One category of classifiers are decision-trees. These classifiers use training data to make a tree structure in which each node in the tree represents a feature, and each leaf represent a class or decision. One of the most well known decision-tree algorithm is J.R. Quinlan's C4.5[20, 21] algorithm, which is implemented in Weka as J48 which will be referred to as J48 from this point on.

Another category of classifiers are often called boosting algorithms, which have been known to improve or boost results by combining multiple weaker classifiers.[2, 6, 26] Boosting algorithms generally run multiple types of weaker classifiers over a dataset, each classifier is then assigned a weight which is generally related to its accuracy. This process is then iterated over multiple times such that the better classifiers increase in weight while worse classifiers are reduced in weight. We chose to use Adaboost (AdaBoostM1 in Weka) since it has shown to be a strong boosting algorithm[2, 6, 26]. All experiments were run using 10-fold cross validation which has been shown to provide good estimates for both Bayesian and decision-tree classifiers[13].

### 3.5 Feature Selection

Feature selection is an important part of the classification process since intelligent feature selection can both improve classifier results and performance. Text classification algorithms like the ones discussed above require a preprocessing stage which converts the text of a message into a vector of words or *features*. The simplest approach is to treat each word encountered as a feature, thus each message becomes a vector of Boolean values, each of which indicates the existence or absence of a feature within a message. While simple, the inefficiency in this approach is twofold. First, the number of unique words which can occur in

a data set can grow quite large (we observed over 1.2 million). Second, not all of the words encountered are a strong indicator as to what class a message should belong to, they are in effect "noise." Thus, the goal of feature selection is to reduce the set of features down to a reasonable sized set which contains only the features that are strong indicators that a message should belong in a specific class. In order to quantify such a value we used *Mutual Information* which measures the dependency between variables and provides the the amount of "information content" of a given feature[1]. In this case, mutual information will determine how strongly a given feature indicates that a message should be classified as spam or ham. Thus in order to determine which features are selected, the mutual information value is computed for each potential feature and only the top K are used.

Additional improvement to feature selection can also be done by removing words that occur very frequently or very infrequently [1, 3, 22, 23]. While words that occur infrequently may be a strong indicator as to a messages class, it can only be applied very few messages, thus loosing much of its value. In order to remove infrequently occurring words, a word must occur in at least 5 messages to be considered a possible feature. On the other hand, words which occur very frequently (e.g. the, is, i) lose much of their value since they occur often in both spam and ham messages and therefore can be considered bad features. We chose to remove words which made up for more than .9% of all of the word occurrences within the dataset, after which there was a steep drop-off. This list of terms, often called a stop list, contained 13 terms in all.

## 4. RESULTS

We experimented with the three different classifiers using various combinations of stemming, stripping HTML, and different numbers of features. Our initial tests were run with only the basic techniques like removing punctuation and a stop list. This trial did not include additional techniques like stemming and tag removal. Figure 1 shows the precision values for the different classifiers with varying numbers of features. This graph indicates that in terms of precision the J48 decision-tree outperforms the Naïve Bayes consistently by over 6%. It also outperforms AdaBoostM1 by as much 8% with 200 features. The results for recall are very much the reverse of those for precision as seen in Figure 2. Naive Bayes had the highest recall of the three reaching 88% while J48 was only able to reach 83%. The results for AdaBoostM1 show what frequently seen in the relationship between precision and recall, when one increases the other decreases. For example, with 400 features AdaBoostM1 precision drops from 81% to 76% in terms of precision, meanwhile it increased from 79% to 89% in terms of recall. This is in effect the filter "loosening" and letting though more messages, while more ham messages get through (increased recall), more spam messages also get through (decreased precision). Overall, the three classifiers did not see significant gains with the addition of more features which indicates that either the maximum amount of information was gained from less than 100 features, or that beyond the top features, the rest of them were less useful in classifying messages.

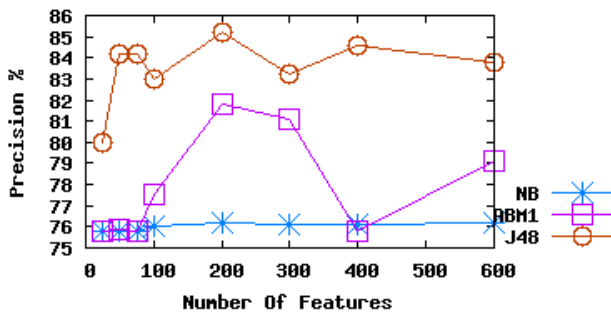


Figure 1.

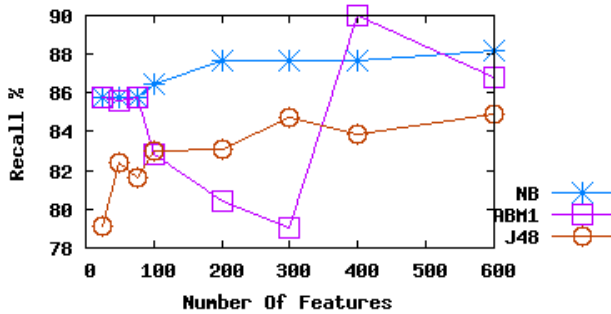


Figure 2.

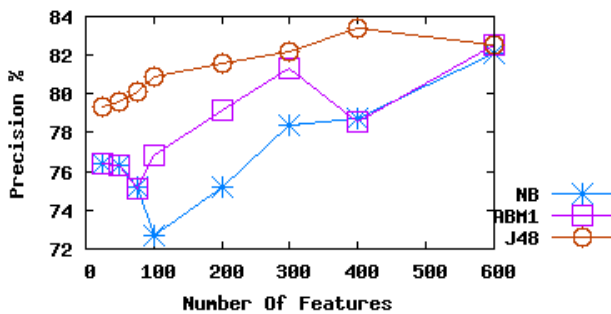


Figure 3.

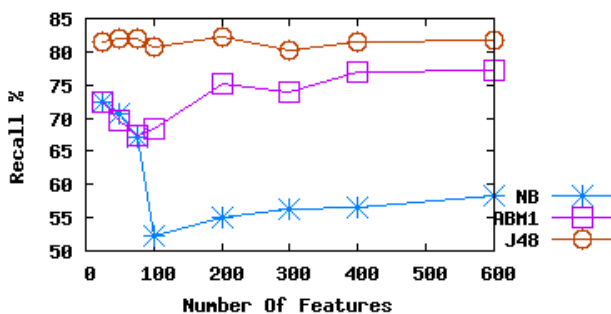


Figure 4.

```

if u want 2 make money Online Enter here &gt; &gt;&gt; <a
href="http://www.facebook.com/group.php?gid=27330495846"
onmousedown="UntrustedLink.bootstrap($(this),
'&quot;964416034bf73447f3beae3939095f79&quot;, event)" target="_blank"
rel="nofollow"><span>http://www.facebook.com/gr</span><wbr></wbr><spa
n class="word_break"></span>oup.php?gid=27330495846</a>

```

Figure 5. Example spam message with tags

Reviewing the features used showed that a large number of them were part of an HTML tag. As seen in the example spam message (Figure 5), links on Facebook are somewhat more complicated than standard links. While removing the tag information would significantly decrease the "noise" in the given message, it would also remove over 2/3 of the message.

Figure 3 shows the results of the classifiers running on the same dataset, except with the tags removed. While the ordering of the classifiers remained the same, there were some notable differences between the two experiments. While J48 was able to reach 83% precision, it required over 400 features to do so whereas previously it was able to attain the same rate with less than 100 features. The recall rate for J48 remained very similar to its previous results. AdaBoostM1 had similar results to the previous trial in that between 100 and 300 features it improved upon Naive Bayes, and beyond that it was fairly similar. The removal of tags from the messages had by far the largest effect on Naive Bayes which was previously unaffected by a larger number of features. While below 100 features the trials are quite similar, it is beyond 100 that Naive Bayes shows a significant increase in precision, even matching that of AdaBoostM1 and J48 at 600 features. As was the case with AdaBoostM1 in the previous trial, gains in precision tend to result in lower recall, and such is the case with Naive Bayes. Beyond 100 features its recall rate dropped below 55% and only slightly recovered. Experiments with the three algorithms beyond 600 features resulted in very little change in precision and recall with 600 features.

A middle ground between the two trials would be to remove the tags from the message, yet retain that information by making tags into features. Experiments with this methodology did not indicate any benefits beyond those of retaining tags.

The addition of stemming in both cases did not result in any large changes for any of the results. Stemming in effect "smoothed" the precision curve for J48 and AdaBoostM1, the dips were slightly increased while the small increases were slightly decreased. None of these changes however resulted in changes larger than a percentage point.

## 4.1 Discussion

A more in depth analysis of the tree created by J48 in its best configuration (no stemming, with tags, 200 features) reveals some interesting information. Even with the 200 features provided to it, only 62 of them are used in the actual tree. Figure 5 shows the CDF for the depth in the tree at which a given message was classified for both correct and incorrect classifications. This figure demonstrates that even of those 62 features included in the tree, most of the classification happened within a depth of 10. Those classifications which happened at the very top of the tree were very unlikely to be correct, this is not terribly shocking since these are mostly messages which contained very few if any of the features and as a result the classifier has very little data to work with. Further depths of the tree also did not prove to be the most accurate for classifying messages; this seems to be the result of over fitting of the training set. This is in line with the fact that the algorithm was able to attain precision and recall above 80% with as few as 50 features.

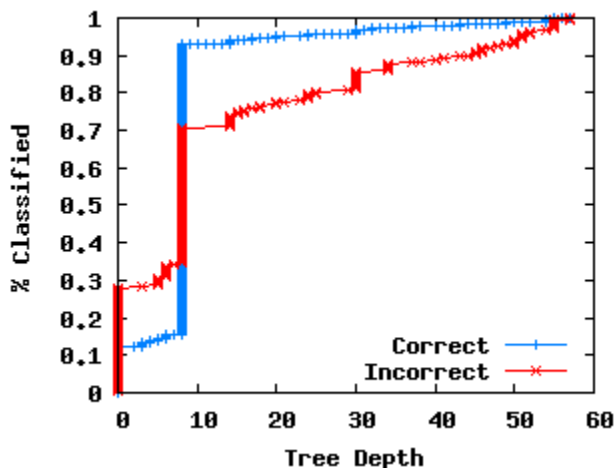


Figure 6. Decision-tree depth of message classification

The performance of Naïve Bayes demonstrated a few of the interesting properties about the algorithm itself. In the case where the tags were left in the messages, the precision and recall remained fairly constant across the number of features. As discussed in [22], Naïve Bayes performs best when features are either completely independent, or functionally dependant. In this case many of the features related to tags are highly dependent since messages containing links are the only ones with tags, yet their “information content” is sufficient such that they are considered a feature. As a result of this the performance of Naïve Bayes is unable to improve. In the second trial Naïve Bayes is able to significantly improve its precision value, but at the cost of its recall.

The performance of AdaBoostM1 is in many ways what one would expect. Given that its classifications are created by combining multiple Bayesian algorithms in the worst case it matches Naïve Bayes, while in the common case it is able to improve upon it. This is most evident in the second trial where the precision of AdaBoostM1 matches or improves upon that of Naïve Bayes, and yet it able to retain high recall where Naïve Bayes flounders.

The

## 5. CONCLUSION

The results of this experiment show that a classifier as simple as J48 can be used effectively to filter spam from Social Networking websites. J48 is able to be quite effective with as few as 50 features and as shown by [10], decision-trees can function at very high rates which is crucial given the amount of messages being sent on a Social Networking any given time. Boosting algorithms like AdaBoostM1 have also shown to be effective for message classification and are able to perform well in cases where a single weak learner did not. While decision-trees have often been passed over for text classification in favor of Bayesian techniques, it may be the case that that decision should be reconsidered, specifically in cases where message lengths are short.

## 6. FUTURE WORK

One avenue for exploration is testing boosting algorithms with decision-trees. Given the strong performance of decision-trees and

boosting algorithms there is a good indicator that the two combined could yield even better results.

While classifying messages we noticed a large number of identical spam messages coming from many different accounts. Looking into this further we found that many of these accounts seemed to be legitimate user accounts with completed profiles, large numbers of friends, and pictures. We suspect that a large amount of the spam on Facebook is actually being created by legitimate accounts which have been stolen by spammers. This is different from what we initially expected. We believed that spammers were creating large numbers of fake accounts in order to spam with, so we expected to find these profiles with very limited information and few friends. Only with further investigation will we be able to verify these suspicions.

## 7. ACKNOWLEDGMENTS

Thank you to the members of the UCSB CURRENT lab for providing us with the Facebook data.

## 8. REFERENCES

- [1] Battiti, R. 1994. Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, vol. 5, pp. 537-550,
- [2] Carreras, X. and Marquez, L. i. 2001. Boosting Trees for Anti-Spam Email Filtering. *CoRR*, vol. cs.CL/0109015,
- [3] Cormack, G. V., et al. 2007. Feature engineering for mobile (SMS) spam filtering. Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, Amsterdam, The Netherlands, 2007.
- [4] Cormack, G. V., et al. 2007. Spam filtering for short messages. Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, Lisbon, Portugal, 2007.
- [5] Facebook. *Facebook Statistics*. Available: <http://www.facebook.com/press/info.php?statistics>
- [6] Freund, Y., et al. 2003. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, vol. 4, pp. 933-969,
- [7] Goodman, J., et al. 2007. Spam and the ongoing battle for the inbox. *Commun. ACM*, vol. 50, pp. 24-33,
- [8] Graham, P. *A Plan For Spam*.
- [9] Jindal, N. and Liu, B. 2007. Analyzing and Detecting Review Spam. Proceedings of the 2007 Seventh IEEE International Conference on Data Mining - Volume 00, 2007.
- [10] Jo, et al. 2003. Accurate decision trees for mining high-speed data streams. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, D.C., 2003.
- [11] Jos, et al. 2006. Content based SMS spam filtering. Proceedings of the 2006 ACM symposium on Document engineering, Amsterdam, The Netherlands, 2006.
- [12] Kanich, C., et al. 2008. Spamalytics: an empirical analysis of spam marketing conversion. Proceedings of the 15th ACM conference on Computer and communications security, Alexandria, Virginia, USA, 2008.

- [13] Kohavi, R. 1995. A study of cross-validation and bootstrapping for accuracy estimation and model selection. 1995.
- [14] Larkey, L. S. and Croft, W. B., "Combining classifiers in text categorization," ed: ACM Press, 1996, pp. 289-297.
- [15] Marilly, E., *et al.* 2002. Service level agreements: a main challenge for next generation networks. In *Universal Multiservice Networks, 2002. ECUMN 2002. 2nd European Conference on*, 2002, pp. 297-304.
- [16] McCallum, A. and Nigam, K. 1998. A comparison of event models for naive bayes text classification. AAAI-98 Workshop on Learning for Text Categorization, 1998.
- [17] Metsis, V., *et al.* 2006. Spam filtering with naive bayes - which naive bayes? Third Conference on Email and Anti-Spam (CEAS), 2006.
- [18] Pantel, P. and Lin, D., "Spamcop A spam classification & organization program," in *In Learning for Text Categorization Papers from the 1998 Workshop*, ed, 1998, pp. 95-98.
- [19] Pantel, P. and Lin, D. 1998. Spamcop: A spam classification & organization program. 1998, pp. 95-98.
- [20] Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning*, vol. 1, pp. 81-106,
- [21] Quinlan, R., *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*: {Morgan Kaufmann}, 1993.
- [22] Rish, I. An empirical study of the naive Bayes classifier. In *IJCAI-01 workshop on "Empirical Methods in AI"*.
- [23] Rogati, M. and Yang, Y. 2002. High-performing feature selection for text classification. Proceedings of the eleventh international conference on Information and knowledge management, McLean, Virginia, USA, 2002.
- [24] Sahami, M., *et al.* 1998. A Bayesian approach to filtering junk email. AAAI Workshop on Learning for Text Categorization, Madison, Wisconsin, 1998.
- [25] Sakkis, G., *et al.* 2001. Stacking classifiers for anti-spam filtering of e-mail. *ArXiv Computer Science e-prints*, June
- [26] Schapire, R. E. *MSRI Workshop on Nonlinear Estimation and Classification, 2002. The Boosting Approach to Machine Learning An Overview*.
- [27] Symantec, "A Monthly Report - August 2007," 2007.
- [28] Witten, I. and Frank, E., *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.
- [29] Yerazunis, W. S. 2004. The Spam-Filtering Accuracy Plateau at 99.9 percent Accuracy and How to Get Past It. In *MIT Spam Conference 2004*, 2004.