Spam Classification of a Social Network

Jon-Michael Deldin
Artificial Intelligence Graduate Project

December 2012

Contents

1 Introduction

Spam is a significant problem in online communities. Spammers advertise their products, services, viruses, and more to members of these social networks. Social networks are targeted because they are generally free, and the spammers can advertise to users via direct messaging or indirect communication (e.g., comments on a post). This severely degrades the legitimate user's

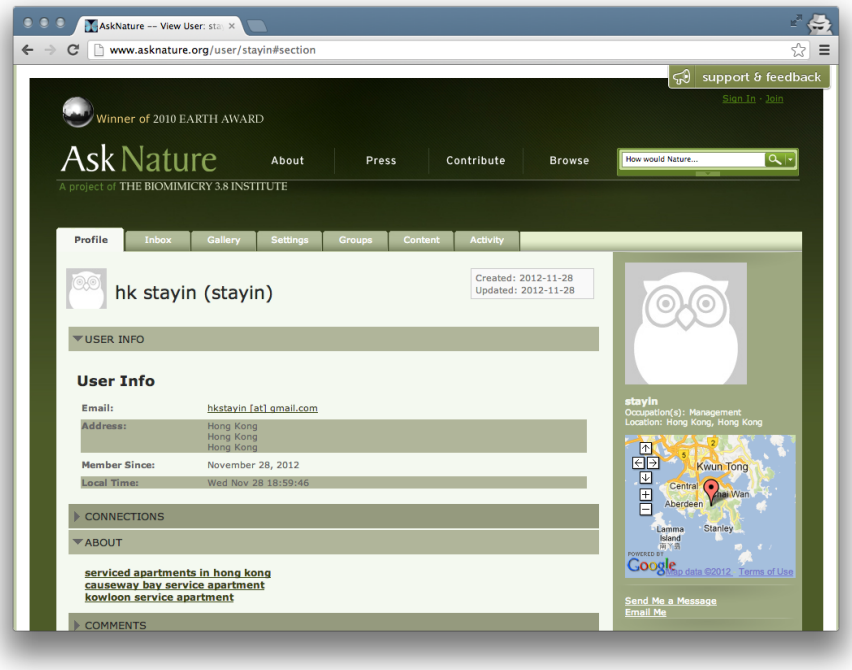

Figure 1: This screenshot of a spam profile shows the ”about” text the spammer enters, along with the first name (”hk”), last name (”stayin”), username (”staying”), and the spammer's address.

experience, affects the website's search engine ranking, and tarnishes the website's public image.

One social network suffering from spam is AskNature[1]. AskNature is a free, online database of biomimetic [1] solutions. Members can create a profile, comment on articles, create and read articles, and participate in forums. Unfortunately, spammers take advantage of the open user registration and inundate the site with illegitimate accounts, such as the one shown in Fig. 1. Since the site opened in November 2008, the staff detected nearly 20,000 spam profiles, in contrast to almost 9,000 legitimate profiles[2]. The site has coped with a few heuristics tools that check for links and HTML, but they are run only to detect profiles – a human still inspects and bans users.

In this project, I will develop a naive Bayes classifier for detecting spam profiles based on the user's biography. The goal will be to determine whether an account is spam or ham (i.e., a legitimate account) based on the plain text of the profile's biography. I hypothesize a unigram bag of words model will

[1] http://www.asknature.org
[2] As of November 28, 2012

outperform (see §3) character gram and phonogram models. Character gram models may suffer from too large of a search space, while phonograms (words represented phonetically) will have too narrow of a search space, leading to increased false positives and false negatives.

2 Naive Bayes Classifier

The research presented here utilizes a naive Bayes classifier to classify spam and ham biographies. These classifiers are very effective for classifying documents [5, p182], and they are widely used in email spam filters [4]. This classifier uses the maximum a posteriori estimate to predict which class a document belongs to. The "naive" assumption is that each attribute is independent of another, which simplifies computing a predicted class v to

$$v = \arg\max_{x \in X} \Pr(x) \prod_i \Pr(a_i|x),$$

where X is the set of possible classes (e.g., spam and ham), and a_i is an attribute (i.e., a term) [5, p177]. The probability of an attribute given a class is often computed as a maximum likelihood estimate (MLE):

$$\Pr(a_i|x) = \frac{\text{COUNT}_x(a_i)}{N_x},$$

where COUNT_x returns the number of occurrences of a_i in the vector of attributes $< a_1, a_2, \ldots, a_d >$ belonging to class x, and N_x is the total number of attributes for class x [5, p177].

MLE is not without its faults. When the number of times an event occurs is small, the probability will be zero, which will overfit the data. Fortunately, additive smoothing (Laplace smoothing) combats this by adding a smoothing parameter α[3]:

$$\Pr(a_i|x) = \frac{\text{COUNT}_x + \alpha}{N_x + \alpha d}$$

In text classification, a bag-of-words model is typically used to capture the frequency of each term appearing in a document. The attributes/terms are words or characters broken up into n-grams, where a unigram corresponds to a single word (e.g., bag = {"not": 1, "secret": 1}), a bigram corresponds to two words (e.g., bag = {"not-secret": 1}), etc.

3 Evaluation

k-fold cross-validation, described in Algorithm 1, is used to compare performance between classifiers. Cross-validation will return a confusion matrix – a table of false positives (ham marked as spam), false negatives (spam marked as ham), true positives (spam), and true negatives (ham)[4]. This matrix results

[3] http://ai-class.com, Unit 5, Topic 21

[4] For brevity, fp will denote a false positive, tn will denote a true negative, and so on.

Algorithm 1 k-fold cross-validation described by [5, p147]

```
function CROSSVALIDATE(data, k)
    partitions ← divide data into k disjoint subsets
    for i ← 1 → k do
        T ← data \ partitions_i
        TRAIN(T)
        CLASSIFY(partitions_i)
        return confusion matrix
    end for
end function
```

in the metrics summarized in Table 1. In this paper, high accuracy is important because it will classify more spam and ham, but it should not come at the cost of low precision (i.e., more legitimate users marked as spammers). Thus, it is unlikely for there to be a clear winner, as it's often a tradeoff between accuracy, precision, and recall. In the event of a tie or unclear winner, the model that is easiest to compute will win, as this classifier will be deployed as a background process on a commodity server.

4 Corpus

The data consist of user biographies – Unicode text fields – spanning 2008-09-03–2012-11-28. These profiles[5] are from AskNature's MySQL database, and the spam ones were manually annotated by AskNature staff during that period. Procuring the corpus involves the following steps:

1. Select non-empty biographies from all active and banned (spam) users
2. Sanitize each biography according to Algorithm 2
3. Save each biography to disk as plain text for easier distribution

[5] "Profile" and "biography" will be used interchangeably from this point forward.

Table 1: Summary of evaluation metrics [3]. Fewer false positives are represented by a higher precision, and fewer false negatives are represented by a higher recall rate.

Metric	Formula	Describes
accuracy	$\frac{tp+tn}{tp+fp+fn+tn}$	correctly predicted classes
precision	$\frac{tp}{tp+fp}$	correctly predicted positive classes
recall	$\frac{tp}{tp+fn}$	how sensitive the classifier is to input

Table 2: Summary statistics for the non-empty biographies processed according to Algorithm 2.

Class	N	Mean Length (words)	σ
Spam	18,498	31.1	85.8
Ham	3,271	61.1	96.2

4. Determine the mean biography length
5. Segment the samples into training and testing sets according to Algorithm 3

4.1 Mean Biography Length

In order to maximize the performance of a classifier, appropriate documents must be selected. Empty samples are obviously not useful in classifying profiles, and samples with one or two words are unlikely to add reliable information. One way to determine a minimum count is to take the mean word counts across spam and ham samples. As shown in Table 2, the data include more spam accounts than ham, and those spam accounts have fewer words per biography on average. The wide standard deviation for both classes is explained by two factors: Users are not required to fill out the biography field, and even if they choose to, no stated length requirements exist. These results indicate the floor of the minimum mean should be used avoid rejecting too much data. Applying a minimum length of 31 words per biography reduces the eligible samples down to 4,702 spam entries and 1,740 ham entries.

5 Methods

Word[6], character, and phonogram models were evaluated with a naive Bayes classifier. For each gram type, the effect of a smoothing parameter of 1 was

[6]A word is defined by any Unicode string containing one or more alphabetic characters. This can be encoded as a case-insensitive Perl Compatible Regular Expression (PCRE): `/[a-z']+/i`.

Algorithm 2 Preprocessing routine for biographies

1. Sanitize all HTML by retaining just the `innerHTML` (the content between start and closing tags). For example, `<a href="http://example.com">link</a>` becomes `link`.
2. Remove all URLs
3. Replace excess whitespace with single spaces

Algorithm 3 Segmenting the data samples into training and test sets

1. Create a list of all profiles and randomize it
2. Select all entries with biographies of 31 or more words
3. For each class (ham and spam), randomly select 100 entries for testing and write to a new `pruned/testing` directory.
4. Write the remaining samples to `pruned/training`

evaluated. Each model is a different way of tokenizing a string. All words were evaluated, including "stopwords" like "the", "is", and "at" in order to retain phrases. In character-grams, a gram is a Unicode character, and grams of length 1-5 were evaluated. As an example of a character gram, the trigrams for "naive" are "nai", "aiv", and "ive". In phonograms, the gram is the phonetic representation of a word. Phonograms were determined by the double-metaphone algorithm [6], which reduced words to 12 consonant sounds.

6 Results

The results of cross-validating and testing word, character, and phonetic models are presented in Tables 3–8.

6.1 Word Grams

Table 3 shows the 10-fold cross-validation results for unigram, bigram, and trigram word models. Trigrams with no smoothing yield the highest accuracy of 93.7%, but the precision is low at 92.1%. In some classifiers, this would be acceptable, but this model would cause too many legitimate users to be banned. The model with both a high accuracy (above 90%) and high precision is the unigram model with $\alpha = 1$. This only resulted in 21 ham profiles being marked as spam, which is much more acceptable than 390+ false positives. While this model is not great at detecting spam in the cross-validation, it actually outperforms the other word-grams on the 200 test samples. Table 4 shows the unigram model with $\alpha = 1$ achieving 94% accuracy and 94.9% precision.

6.2 Character Grams

Table 5 reports the 10-fold cross-validation matrix and evaluation metrics for character grams of length 1-5[7] The best performing character-gram was a 4-

[7]Character grams longer than five characters were not pursued after disappointing results in an earlier experiment – accuracy and precision decrease after four characters, and computing cross-validation matrices for larger grams takes a significant amount of computing power.

Table 3: 10-fold cross-validation confusion matrix for word-gram models ($N = 6442$)

tokenizer	α	accuracy	precision	recall	tp	tn	fp	fn
unigram	0	0.926	0.919	0.986	4536	1241	399	66
unigram	1	0.905	0.995	0.876	4032	1619	21	570
bigram	0	0.936	0.921	0.999	4599	1246	394	3
bigram	1	0.563	0.996	0.409	1884	1632	8	2718
trigram	0	0.937	0.921	0.999	4599	1247	393	3
trigram	1	0.405	0.997	0.194	893	1637	3	3709

gram with $\alpha = 1$ for a smoothing parameter. This model achieved 93.2% accuracy and 99.2% precision in cross-validation, and 92.5% accuracy and 95.7% precision on the test samples (Table 6). The 5-gram with $\alpha = 1$ model also achieved the same results on the test samples, but given it's cross-validation performance and the cost to compute, it is not the best character gram.

6.3 Phonograms

Phonograms performed very poorly, as evidenced by Table 7 and 8. The best model was a single phonogram with no smoothing, which achieved 79.5% accuracy and 85.1% precision in cross-validation, but 61% accuracy and precision in testing. The two-phonogram model fared poorly with and without smoothing as well. It had 100% recall, which came at the cost of 1,640 false positives.

7 Discussion

Of the best performing gram types, summarized in Table 9–10, both the 1-word gram and 4-character gram have high accuracy and precision. The word gram has higher accuracy in testing with only one more false positive than the character gram. However, according to the cross-validation results and the evaluation metrics described in §3, the 1-word gram is superior due to it's

Table 4: Word-gram confusion matrix for $N = 200$ test samples

tokenizer	α	accuracy	precision	recall	tp	tn	fp	fn
unigram	0	0.670	0.840	0.420	42	92	8	58
unigram	1	0.940	0.949	0.930	93	95	5	7
bigram	0	0.540	1	0.080	8	100	0	92
bigram	1	0.700	1	0.400	40	100	0	60
trigram	0	0.530	1	0.060	6	100	0	94
trigram	1	0.600	1	0.200	20	100	0	80

"good enough" accuracy and highest precision, and it is easy to compute. This supports the hypothesis, but not clearly since character grams are very close. Phonograms had low precision and recall, which also supports the hypothesis that representing words phonetically will make the spam and ham vocabularies too similar.

8 Conclusion

This paper developed a naive Bayes classifier for the AskNature social network that classifies spam profiles with 94% accuracy and 93% precision with a unigram word model. Both unigram word and 4-character gram models had high accuracy and precision in 10-fold cross-validation of 6442 samples and testing on 200 samples. Phonograms performed poorly as expected.

Future experiments may include combining the classifier developed here with another classifier or heuristic to form an ensemble [2]. A weighted approach, where if one classifier returns a probability with some uncertainty, the second classifier will run, and the maximum probability will result in the prediction. This should reduce false positives and false negatives. Another useful experiment would be on-line, continuous learning, where the classifier is trained every time AskNature's staff tag a spam or ham profile. Finally, varying the spam-to-ham ratio, as described in Appendix A may be worth investigating to see what ratio of

A Determining the Ratio of Spam to Ham

Algorithm 4 describes a method of evaluating all possible spam-to-ham ratios from 1% to 99%, with the caveat that the best ratio is influenced by a high accuracy and as high as possible precision. This algorithm requires balanced training sets, which would substantially reduce the number of spam entries. AskNature adds ~300 spam and ~200 ham accounts each month, so this algorithm can be run once there are more ham accounts with longer biographies.

B Implementation

The entire project is developed as a set of Ruby 1.9.3 and R scripts that are glued together by a `Rakefile` for automation. The data files are distributed as plain text files in the `src/data/processed` directory, so a MySQL database is not needed to classify text. Since they need to be accessed often, I have serialized them and saved them to disk for easy loading in `src/data/objects`.

The following open-source libraries are used:

- Ankusa[8], an implementation of naive Bayes for text classification. I extended it in a number of ways. In terms of performance, I sped up word

[8] https://github.com/livingsocial/ankusa

Algorithm 4 Determine the ratio of spam:ham messages with the greatest accuracy.

```
function FINDBESTRATIO(samples, n)                    ▷ Takes a set of n samples
    spam ← SHUFFLE(samples \ ham)    ▷ Randomizes the order of samples
    ham ← SHUFFLE(samples \ spam)
    best-accuracy ← 0
    best-ratio ← ∅

    for i ← 0.0 → 1.0, step ← 0.01 do
        ham-ratio ← ni
        spam-ratio ← n(1 − i)
        limited-samples ← TAKE(ham, ham-ratio) ∪ TAKE(spam, spam-ratio)

        CROSS-VALIDATE(limited-samples)
        if accuracy > best-accuracy then
            best-accuracy ← accuracy
            best-ratio ← {ham-ratio, spam-ratio}
        end if
    end for
    return best-ratio, best-accuracy
end function
```

hashing and string conversion, resulting in 3-6 s improvement per classification, which adds up when cross-validating multiple models. For features, I implemented configurable Laplace smoothing and added n-gram support for word, character, and phonetic grams. I intend to contribute these changes upstream as soon as possible.

- Text[9], for the Double Metaphone implementation.
- Parallel[10] for simplified multi-core and multi-threaded computing. My cross-validation routine is fully parallelized and uses six cores (configurable). My database import scripts are threaded, as they write thousands of files to disk. Wherever possible, I tried to parallelize my code because classifying text is really CPU intensive.
- MySQL and Sequel[11] for accessing a database and fetching records.
- Sanitize[12] for removing HTML.

Please consult `src/README.md` for more details.

[9] http://text.rubyforge.org
[10] https://github.com/grosser/parallel
[11] http://sequel.rubyforge.org
[12] https://github.com/rgrove/sanitize

References

[1] Janine M Benyus. *Biomimicry: innovation inspired by nature*. Morrow, New York, 1st edition, 1997.

[2] N. Chawla, S. Eschrich, and L.O. Hall. Creating ensembles of classifiers. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 580 –581, 2001.

[3] Charles Elkan. Evaluating classifiers. http://cseweb.ucsd.edu/~elkan/250B/classifiereval.pdf, 01 2012.

[4] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes – which naive bayes? In *Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[5] Tom M Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[6] Lawrence Phillips. The double metaphone search algorithm. http://www.drdobbs.com/the-double-metaphone-search-algorithm/184401251?pgno=2, June 2000.

Table 5: 10-fold cross-validation confusion matrix for character-gram models ($N = 6442$)

tokenizer	α	accuracy	precision	recall	tp	tn	fp	fn
1	0	0.696	0.882	0.679	3124	1221	419	1478
1	1	0.696	0.882	0.679	3125	1221	419	1477
2	0	0.855	0.948	0.851	3916	1424	216	686
2	1	0.862	0.969	0.840	3867	1516	124	735
3	0	0.881	0.901	0.943	4339	1162	478	263
3	1	0.926	0.986	0.912	4198	1580	60	404
4	0	0.911	0.900	0.990	4556	1131	509	46
4	1	0.932	0.992	0.915	4213	1605	35	389
5	0	0.931	0.915	0.999	4598	1215	425	4
5	1	0.897	0.865	3979	1619	21	623	

Table 6: Character-gram confusion matrix for $N = 200$ test samples

tokenizer	α	accuracy	precision	recall	tp	tn	fp	fn
1	0	0.710	0.728	0.670	67	75	25	33
1	1	0.710	0.728	0.670	67	75	25	33
2	0	0.830	0.837	0.820	82	84	16	18
2	1	0.865	0.884	0.840	84	89	11	16
3	0	0.800	0.806	0.790	79	81	19	21
3	1	0.915	0.919	0.910	91	92	8	9
4	0	0.645	0.738	0.450	45	84	16	55
4	1	0.925	0.957	0.890	89	96	4	11
5	0	0.580	0.900	0.180	18	98	2	82
5	1	0.925	0.957	0.890	89	96	4	11

Table 7: 10-fold cross-validation confusion matrix for phonograms ($N = 6442$)

tokenizer	α	accuracy	precision	recall	tp	tn	fp	fn
unigram	0	0.795	0.851	0.875	4029	932	708	573
unigram	1	0.558	0.737	0.623	2865	620	1020	1737
bigram	0	0.737	0.737	1	4602	0	1640	0
bigram	1	0.737	0.737	1	4602	0	1640	0

Table 8: Phonogram confusion matrix for $N = 200$ test samples

tokenizer	α	accuracy	precision	recall	tp	tn	fp	fn
unigram	0	0.610	0.610	0.610	61	61	39	39
unigram	1	0.610	0.610	0.610	61	61	39	39
bigram	0	0.500	0.500	1	100	0	100	0
bigram	1	0.500	0.500	1	100	0	100	0

Table 9: 10-fold cross-validation comparison of the best performing gram models ($N = 6442$)

tokenizer	α	accuracy	precision	recall	tp	tn	fp	fn
1-word	1	0.905	0.995	0.876	4032	1619	21	570
4-char	1	0.932	0.992	0.915	4213	1605	35	389
1-phono	0	0.795	0.851	0.875	4029	932	708	573

Table 10: Comparison of the best performing models on $N = 200$ test samples

tokenizer	α	accuracy	precision	recall	tp	tn	fp	fn
1-word gram	1	0.940	0.949	0.930	93	95	5	7
4-char gram	1	0.925	0.957	0.890	89	96	4	11
1-phono gram	0	0.610	0.610	0.610	61	61	39	39