Patrick Vacek
Final Project
MATH 3220

Towards Better Bayesian Spam Filters

Executive Summary

Spam, that is, unsolicited commercial or bulk email, is a rising problem that has become unavoidable to nearly all email users, and automatic filtering has become a strong area of interest for many users. The most widespread and effective technique for doing so is to use a naïve Bayesian classifier. This technique uses Bayes' theorem to determine the probability that the presence of each word in a given email corresponds to the email being spam or non-spam based on evaluation of previous spam and non-spam emails and analysis of their contents. By combining these probabilities together, the probability that a newly received email is spam or non-spam can be calculated, and filtering can be applied as necessary.

The Bayesian filter approach has several advantages: it customizes itself to an individual's behavior, it continuously improves its performance, and it adapts to new spam techniques on its own (Graham, 2002). However, no filter is perfect, and every filter will accidentally allow some spam through (false positives) and occasionally mark a legitimate email as spam (false negatives). False positives are quite undesirable, and thus the filtration algorithm should be optimized to greatly reduce these while still allowing for few false negatives.

I will test the effectiveness of multiple Bayesian algorithms on a spam dataset and several variations thereof to find an ideal filtration configuration. I will use a spam dataset created from 4601 emails received in 1997 by a Hewlett-Packard Labs employee (Hopkins, 1999). Each email is described by 58 attributes, of which 54 are numeric values that indicate how often a certain word or symbol appeared in the email, three are analyses of capital letter frequencies, and the last is the predetermined class (spam or non-spam).

I used eight variations of the dataset in addition to the unaltered original. The first replaces the capital letter length analyses with the logarithms of these values, and the second entirely removes these analyses. The third rounds all of the values in the original dataset up the the nearest whole number using a ceiling function. The fourth also uses the ceiling function but additionally replaces the length analyses with logarithmic values, and the fifth uses the ceiling function and removes the length analyses. The sixth variation converts the word frequencies into boolean values of whether or not the word appeared in the email or not. The seventh uses the boolean values and logarithmic length analyses, and the eighth uses the boolean values and removes the length analyses.

I used Weka to process six algorithms: a basic naïve Bayesian classifier, a multinomial variation (which works with continuous data better than the basic version), a Bayesian network classifier, a hidden naïve Bayesian (HNB) classifier, an averaged one-dependence estimator (AODE), and a weighted AODE (WAODE). The multinomial classifier is designed to work better with continuous data than the normal naïve classifier, but it cannot work with boolean or entirely categorical data. The latter three algorithms are all significantly more complex and require entirely categorical data. Since the dataset variations using the ceiling or boolean values combined with either logarithmic length analyses or the absence of these analyses are the only variations that are entirely categorical, only those four could use the more advanced algorithms.

Each algorithm was applied to each variation three times, using a different ordering of the dataset to create a training and test set. Each training set was used to build a model to apply to the related test set, and the performances across the three were averaged to create the following chart, which represents what percentage of spam emails were correctly classified in the test datasets by each algorithm:

	Naïve	Multinomial	Net	HNB	AODE	WAODE
Unaltered	79.13%	77.43%	89.43%	X	X	X
Length Logarithms	79.00%	86.33%	89.60%	X	X	X
No Lengths	78.60%	86.73%	89.50%	X	X	X
Ceiling	89.77%	79.83%	88.40%	X	X	X
Ceiling / Logarithms	88.32%	X	88.36%	91.82%	91.32%	93.32%
Ceiling / No Lengths	88.50%	X	88.90%	91.43%	90.93%	92.50%
Boolean	85.08%	X	88.53%	X	X	X
Boolean / Logarithms	88.23%	X	88.30%	90.60%	92.17%	92.37%
Boolean / No Lengths	88.30%	X	88.33%	90.53%	91.03%	91.90%

The WOADE algorithm as applied to the ceiling function variation with logarithmic length analyses had the best performance at 93.32%. However, this is only a few percentage points better than the performance of all of the advanced algorithms on all of the entirely-categorical variations. Therefore, other unrelated datasets may find that other variations or algorithms may outperform the optimal choice here. It should be noted that 93.32% is a fairly poor performance for a spam filter, but if the dataset was expanded to include every word in every filter, this performance would likely improve dramatically.

Problem Description

Spam, also known as junk email, is a rising problem that has become unavoidable to nearly all email users. It is usually defined as unsolicited commercial or bulk email, but it is sometimes simplified to just any unwanted email (E-mail spam, 2008). Graham (2002) has pointed out that spam is not actually necessarily unsolicited or commercial, and he instead defined spam as unsolicited automated email. Regardless, individuals, businesses, and ISPs all have found it desirable to filter spam by automated means. Initial attempts to do so began by simple rulesets, which usually operate by using certain words or phrases as a strict indication of whether an email was spam or not (Cranor, 1998). However, if these rulesets are static and publicly available (as is usually the case with most commercial software), a spammer could simply work around the rulesets (Bayesian spam filtering, n.d.). Furthermore, such rulesets are language-dependent and not specific to an individual or company's behavior (Ibid.).

The simple, ultimately ineffective, solution is to simply expand the ruleset to cover multiple languages and counteract new spammer techniques, but doing so means the spammers will simply adapt to the new protections with new tricks (Bayesian spam filtering, n.d.). A superior solution, first popularized by Graham (2002), is to using naïve Bayesian statistics to probabilistically determine the likelihood that an email is spam. Graham's technique is to apply Bayes' theorem to each word or token (numbers, non-alphanumeric characters, etc.) to determine how likely it is that an email containing that word will be spam. Bayes' theorem can be represented mathematically as follows:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

(Bayes' theorem, 2008). This calculation must be calculated twice for each word to determine that probability that it corresponds to spam and to non-spam. P(A) represents the probability of a given class (spam or non-spam) appearing in the entire set of emails. P(B) represents the probability that the given word appears in the entire set of emails. P(B|A) is the probability that the given word is found in emails that are either spam or non-spam. P(A|B) will thus be the probability that an email is spam or non-spam if the given word is found in the email. Of course, to begin this process, there must already be an established core of data containing the content of a user's emails and a manual evaluation of each email as spam or non-spam (Graham, 2002). When a new email is received, the likelihoods that each word in the email correspond to spam (that is, the P(A|B) values where A is spam) are multiplied together. The same is done for non-spam likelihoods. Often, only the spam likelihood is used, and if the spam likelihood is greater than a determined threshold (i.e., .95), the email is classified as spam. However, different applications may use combinations of the two probabilities to make final judgments.

The advantages to this method are plentiful. Because of its very nature, Bayesian filtering is customized to an individual's personal behavior and email activity (Graham, 2002). Language is a non-issue, and the filter will continuously train and improve itself if the user reviews and corrects the filter's activity (Ibid.). Even as spammers develop new techniques, the Bayesian filter will simply adapt on its own (Ibid.).

However, no filter is perfect, and any filter will accidentally allow some spam through and occasionally mark a legitimate email as spam. False negatives (spam that gets past the filter) are generally considered not as egregious of a fault as false positives (marking legitimate email as spam), probably because users are less likely to notice when an email was not properly received than when an undesirable email made it past the filter. (Graham, 2002; The grumpy editor's guide, 2006). Fine-tuning to minimize false positives is thus essential. The question thus becomes what sort of modifications to the algorithm should be used to optimize the filter performance.

In my own previous research, I compared the effectiveness of using a naïve Bayesian classifier against four other algorithms (and an aggregate determined by voting) to classify spam (Vacek, 2008). In that case, I found that the naïve Bayesian classifier performed worse than every other method. However, in contemporary discussions of spam filtering, Bayesian techniques are nearly ubiquitous (see Bayesian spam filtering, n.d.; Bayesian spam filtering, 2008; Dumais, 1998; Graham, 2002; The grumpy editor's guide, 2006; etc.). Therefore, I will continue research on the same dataset by making modifications to the dataset and by using multiple Bayesian classification algorithms to find a more optimal performance outcome.

The dataset was created from 4601 emails received in 1997 by a Hewlett-Packard Labs employee named George Foreman (Hopkins, 1999). Each email is described by 58 attributes, of which 54 are numeric values that indicate how often a certain word or symbol appeared in the email (specifically, 100 * number of times the word or symbol appeared in the email / number of words or symbols in the email). The words analyzed cover a broad spectrum of business terms ("report", "telnet", etc.), words that often appear in spam ("credit", "free", etc.), terms relevant only to the specific user ("george", "hpl", etc.), and everything in between. Three of the attributes concern capital letter frequencies: average length of uninterrupted sequences of capital letters, longest length of an uninterrupted sequence of capital letters, and the total number of capital letters in the email. The final attribute is the author's predetermined classification of the email as spam or non-spam.

Analysis Technique

I decided to use Weka as my primary research and analysis tool because of its ease of use and the large number of algorithms that are built in to it. I found six Bayesian algorithms to test the dataset with, and I made several sets of modifications to the data to try to find if certain algorithms would work best with certain adjustments. I chose to create eight variations of the dataset (in addition to the unaltered original) to try out several options. The default, unaltered dataset has the standard 54 word frequencies, the three character sequence length analyses, and the predetermined classification. All 57 non-class attributes of the standard set are continuous.

The first variation substitutes the logarithms of the length analyses for their normal values. Since the range of values for these attributes is from 1 to 15841, the logarithms of these values would bring the range to 0 to 5, which is much closer to the range of the frequency attributes (of which the largest range is 0 to 42.81 for the word "3d"). I created another variation by simply removing the length analyses from the data entirely, since most basic spam filters do not use these measurements (for example, Graham (2002) does not mention these sequence lengths).

Since some algorithms require the use of only categorical data, I decided to experiment with changing the real-valued frequencies and length analyses into categorical data. The third variation rounds all the numbers up to the nearest whole number by means of a ceiling function. I decided against rounding down (truncation) or standard rounding because it is necessary to separate the event of a word not appearing at all in email from a word appearing in an email but infrequently enough that its relative frequency would round down to zero. Many spam filters function based solely on whether or not a word appears at all, not on how frequently it appears. This allows for turning each of the frequency attributes into categorical attributes (where each category is a whole number value), but since the length analyses were left to remain as continuous attributes, this variation will still not cooperate with certain algorithms that require entirely categorical data.

The fourth variation takes the ceiling function version and also replaces the length analyses with their logarithmic counterparts (rounded up to the nearest whole number). Since the range of these rounded logarithm values is just zero to five, they were easily turned into categorical data. Another variation based off the ceiling function version just removes the length analyses entirely to be left with an entirely categorical dataset.

Since many filters operate based on whether or not words appeared instead of relative frequency in the email, I decided to go a step further than just separating zero from numbers that could round down to zero. The sixth variation I created replaces all the frequency attributes with simple boolean values, where zero corresponds to an original value of zero and one corresponds to anything greater than zero in the original dataset. The length analyses were left alone as continuous data.

The next variation is based off the boolean version, but the length analyses are replaced by their logarithmic relatives (again rounded up to the nearest whole number) to reduce their range and to transform them into categorical data. The eighth and final variation is also based off the boolean version but removes the length analyses entirely.

For each variation (and the original dataset), I randomized the data instances three times to create three different orderings of the data. In each case, I separated the datasets into 3601 instances for use in training a model and 1000 instances to be tested against that model. By having three different orderings of the data for each variation, I can use the algorithms to create slightly different models based on the instances including in the training set. The performance of each model against its appropriate test set can then be averaged to yield a more precise final evaluation of the accuracy of spam detection based on a combination of data modification and algorithm choice.

The six algorithms from Weka that I chose to apply to each dataset variation are a basic naïve Bayesian classifier, a multinomial variation (which works with continuous data better than the basic version), a Bayesian network classifier, a hidden naïve Bayesian (HNB) classifier, an averaged one-dependence estimator (AODE), and a weighted AODE (WAODE). Weka does not provide detailed explanations of these algorithms and their functionality, but the source code is publicly available (see Frank, 2007). The former three algorithms are relatively simple, while the latter three are more complex and take noticeably longer to compute. The multinomial classifier requires that at least one attribute is continuous (or else it would presumably function no different than the normal version), so the dataset variations that are purely categorical cannot use the algorithm. (It also cannot handle boolean attributes.) Similarly, the latter three algorithms require continuous data, so the original dataset and the variations with continuous attributes cannot take advantage of these algorithms.

Assumptions

I assumed that the Weka algorithms that I chose approximate Bayesian techniques that actual spam filters use. Since explanations of the algorithms are minimal, I am left to assume that they function as I expect them to.

If new instances were added to the dataset, the categories used for the frequencies and length analysis logarithms may have to be expanded due to new, extreme cases, which could reduce overall performance. I have assumed that this will not happen, and the categories chosen for each attribute will remain static.

Results

The following chart indicates what percentage of each test set variation was correctly classified by each algorithm:

	Naïve	Multinomial	Net	HNB	AODE	WAODE
Unaltered 1	80.00%	77.50%	90.30%	X	X	X
Unaltered 2	79.40%	79.70%	89.20%	X	X	X
Unaltered 3	78.00%	75.10%	88.80%	X	X	X
Length Logarithms 1	79.70%	86.90%	90.30%	X	X	X
Length Logarithms 2	78.90%	86.50%	89.20%	X	X	X
Length Logarithms 3	77.80%	84.20%	88.90%	X	X	X
No Lengths 1	79.60%	87.70%	90.00%	X	X	X
No Lengths 2	78.90%	87.40%	89.20%	X	X	X
No Lengths 3	77.30%	85.10%	89.30%	X	X	X
Ceiling 1	90.60%	80.20%	89.20%	X	X	X
Ceiling 2	90.80%	82.00%	88.20%	X	X	X
Ceiling 3	87.90%	77.30%	87.80%	X	X	X
Ceiling / Logarithms 1	88.62%	X	88.62%	91.92%	92.12%	93.61%
Ceiling / Logarithms 2	88.22%	X	88.22%	92.52%	92.22%	94.64%
Ceiling / Logarithms 3	88.12%	X	88.22%	91.02%	89.62%	91.72%
Ceiling / No Lengths 1	88.90%	X	89.30%	91.80%	91.30%	92.50%
Ceiling / No Lengths 2	88.50%	X	89.10%	91.90%	91.50%	93.40%
Ceiling / No Lengths 3	88.10%	X	88.30%	90.60%	90.00%	91.60%
Boolean 1	84.50%	X	87.90%	X	X	X
Boolean 2	85.25%	X	88.67%	X	X	X
Boolean 3	85.48%	X	89.03%	X	X	X
Boolean / Logarithms 1	88.50%	X	88.60%	90.50%	92.60%	92.80%
Boolean / Logarithms 2	88.30%	X	88.40%	91.50%	93.50%	93.80%
Boolean / Logarithms 3	87.90%	X	87.90%	89.80%	90.40%	90.50%
Boolean / No Lengths 1	88.40%	X	88.40%	90.40%	91.10%	92.10%
Boolean / No Lengths 2	88.80%	X	88.80%	91.50%	91.80%	93.20%
Boolean / No Lengths 3	87.70%	X	87.80%	89.70%	90.20%	90.40%

The data set variations are listed in the order described above in the Analysis Technique; each one has three different versions based on the random reorderings of the data to mix up which data instances were used in training or in testing. "Naïve" represents the naïve Bayesian classifier, "Multinomial" represents the naïve multinomial classifier, "Net" represents the Bayesian network classifier, "HNB" represents the hidden naïve Bayesian classifier, "AODE" represents the averaged one-dependence estimator, and "WAODE" represents the weighted AODE. Xs indicate that the algorithm could not be applied to the associated dataset variation.

To simplify the data and make it easier to compare across data and algorithm variances, the following chart displays the average values of each set of three versions from each dataset variation:

	Naïve	Multinomial	Net	HNB	AODE	WAODE
Unaltered	79.13%	77.43%	89.43%	X	X	X
Length Logarithms	79.00%	86.33%	89.60%	X	X	X
No Lengths	78.60%	86.73%	89.50%	X	X	X
Ceiling	89.77%	79.83%	88.40%	X	X	X
Ceiling / Logarithms	88.32%	X	88.36%	91.82%	91.32%	93.32%
Ceiling / No Lengths	88.50%	X	88.90%	91.43%	90.93%	92.50%
Boolean	85.08%	X	88.53%	X	X	X
Boolean / Logarithms	88.23%	X	88.30%	90.60%	92.17%	92.37%
Boolean / No Lengths	88.30%	X	88.33%	90.53%	91.03%	91.90%

There a plenty of observations to be made from this chart. The basic naïve Bayesian classifier is the least effective algorithm, while the multinomial version only improves results if the length analyses are simplified or removed. The network classifier performs equally or better in the case of the fully-continuous variations. Transforming the frequencies into categorical data improves the performance of the naïve classifier, but slightly reduces the effectiveness of the network classifier. The three more advanced algorithms that require categorical data all consistently outperform the simpler algorithms.

The best performance is therefore achieved by applying the ceiling function to all frequencies and to the logarithms of the length analyses and applying the WAODE algorithm. However, since this is only a small number of percentage points better than the performances of all three advanced algorithms on all four entirely categorical dataset variations, the best combination in this case cannot be guaranteed to be the best choice for an unrelated dataset or for spam in general.

Issues

The spam dataset used for this research is fairly limited in application and does not entirely allow for the optimal filtering that is possible. This is because the dataset only lists 54 particular words and tokens to analyze. Ideally, a spam filter would have access to the appearances of every individual word in an email. This would create a massive dataset, but this is necessary to catch the small quirks of spam and non-spam and also to be able to evolve as spam evolves. Although this dataset favors frequencies over boolean representations of a word's appearance, unlike most filters, it is not necessarily bad to have too much information, and it can be converted into boolean data anyway, as I have done.

Since the dataset lacks a complete list of all the words, there is no hope that a filter modeled off of the given attributes could perform as well as necessary. The best performance with the given models was merely 93.32%, which is fairly poor for a spam filter. Due to the given limitations, this is not useless, but most spam filters are able to capture over 99% of spam (Graham, 2002).

Appendix

Beyond Bayesian filtering, there are advanced filtering techniques that do not use Bayes' theorem, but they have achieved minimal popularity thus far (for example, see Markovian discrimination, 2008). There are, however, many methods that complement a filter (be it Bayesian or not) to reduce spam intake, including the following:

- only accept incoming emails that have been authenticated in some fashion
- filter out emails whose checksums match those of known spam kept in a database
- create a blacklist to block specific servers or IP addresses
- create a whitelist of trusted sources that can automatically bypass filters
- reject emails that do not properly follow email protocols
- use a basic ruleset based on specific words or regular expressions (Anti-spam techniques, 2008).

References

Anti-spam techniques (e-mail). (2008). In *Wikipedia, The Free Encyclopedia*. Retrieved December 15, 2008, from http://en.wikipedia.org/w/index.php?title=Anti-spam_techniques_(e-mail)&oldid=256281045.

Bayes' theorem. (2008). In *Wikipedia, The Free Encyclopedia*. Retrieved December 14, 2008, from http://en.wikipedia.org/w/index.php?title=Bayes%27_theorem&oldid=256085354.

Bayesian spam filtering. (n.d.). *AllSpammedUp.com*. Retrieved December 14, 2008, from http://www. allspammedup.com/anti-spam/bayesian-spam-filtering/.

Bayesian spam filtering. (2008). In *Wikipedia, The Free Encyclopedia*. Retrieved December 15, 2008, from http://en.wikipedia.org/w/index.php?title=Bayesian_spam_filtering&oldid=251522184.

Cranor, L.F., and LaMacchia, B.A. (1998). Spam! *Communications of the ACM, 41*(8). Retrieved December 14, 2008, from the ACM Digital Library database.

Dumais, S., et al. (1998). A Bayesian approach to filtering junk e-mail. *Stanford University Computer Science Department*. Retrieved December 14, 2008, from http://robotics.stanford. edu/users/sahami/papers-dir/spam.pdf.

Dunham, M.H. (2003). *Data mining: Introductory and advanced topics*. Upper Saddle River, NJ: Pearson Education, Inc.

E-mail spam. (2008). In *Wikipedia, The Free Encyclopedia*. Retrieved December 14, 2008, from http://en.wikipedia.org/w/index.php?title=E-mail_spam&oldid=258087615.

Frank, E., et al. (2007). *Weka 3: Data Mining Software in Java*. The University of Waikato. Retrieved April 26, 2008, from http://www.cs.waikato.ac.nz/ml/weka/.

Graham, P. (2002). *A plan for spam*. Retrieved December 14, 2008, from http://www.paulgraham.com/ spam.html.

The grumpy editor's guide to Bayesian spam filters. (2006). *Eklektix, Inc.* Retrieved December 14, 2008, from http://lwn.net/Articles/172491/.

Hopkins, M., et al. (1999). *Spam e-mail database*. Hewlett-Packard Labs. Retrieved April 22, 2008, from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/.

Markovian discrimination. (2008). In *Wikipedia, The Free Encyclopedia*. Retrieved December 16, 2008, from http://en.wikipedia.org/w/index.php?title=Markovian_discrimination&oldid= 226348624.

Vacek, P. (2008). Evaluating the effectiveness of a combination of multiple classifiers. Retrieved December 14, 2008, from http://mercury.webster.edu/aleshunas/Support%20Materials/Analysis/Final%20Project%20Vac ek.doc.