DalTREC 2005 Spam Track: Spam Filtering using N-gram-based Techniques

Vlado Kešelj, Evangelos Milios, Andrew Tuttle, Singer Wang,
Roger Zhang
Faculty of Computer Science
Dalhousie University, Halifax, Canada
{vlado,eem,swang,rogerz}@cs.dal.ca

February 10, 2006

Abstract

We briefly describe DalTREC 2005 Spam submission. DalTREC is the TREC research project at Dalhousie University. Four packages were submitted and they resulted in a median performance. The results are interesting and may be seen positive in the light of simplicity of our approaches.

1 Introduction

2005 is the second year that Dalhousie University participated actively in TREC. The project DalTREC[1] was initiated in 2003 and serves as an umbrella for different TREC-related activities at Dalhousie. The tracks of interest were Question Answering, Spam, HARD, Genomics, and Enterprise. We submitted runs for the Question Answering (QA) track and the Spam track. This paper discusses the Spam track.

2 Background Review

Spam, sometimes more formally known as "Unsolicited Commercial Email" (UCE) or "Unsolicited Bulk Email" (UBE), has grown from being a minor nuisance into a global menace. Millions of email users world-wide waste untold thousands of hours every day sorting through incoming mailboxes cluttered with unwanted or even offensive messages. Meanwhile, the growth in network resources consumed by spam continues at an unsustainable rate. The spiralling direct and indirect costs of spam threaten the reliability and utility of the email infrastructure that has become so vital to the global economy.

At the core of the spam problem is the fact that spam allows vendors to transfer the vast majority of their marketing costs to the people they are marketing to. Even though the response rate to spam is believed to be vanishingly small, it is still large enough to overcome the even smaller costs of sending spam. Therefore, it becomes a volume-driven business—the more spam you send, the more money you make. But the real cost of the ensuing flood of junk email is not small, and it is borne by the owners of the networks and servers that must deliver the spam, and by the individual recipients who must waste valuable time sorting through it.

[1] http://www.cs.dal.ca/˜trecacct — DalTREC URL

A variety of legal, structural, and technical solutions have been proposed; most are controversial and many suffer from glaring weaknesses. Legal solutions face daunting jurisdiction problems. Structural solutions, such as replacing SMTP, face staggering infrastructure costs. Many technical solutions are extremely high-maintenance, resulting in a never-ending "arms race" between the elusive spammers and the network administrators attempting to find new ways to block them.

While the world searches for a long-term solution, there is an urgent need for something businesses can use to protect their networks from the escalating costs of spam. This research is motivated by that need.

Example: "Valid-channel" spam. Classifying certain kinds of spam may be very subjective: some people will classify these messages as spam, which some other will classify them as *ham,* i.e., non-spam messages. An example is spam that we call "valid-channel" spam. Namely, some spam messages sneak through e-mail list servers and get distributed to subscribers. By content, this is clearly a spam message, but some user may decide to classify it as ham since it came through an important list form. If the e-mail server itself, removed the message as a spam, that would be correct action, but since the spam message made it through it is not to be ignored any more. This approch would be consistend with the use of white-lists, for example.

3 The Naïve Bayes Classifier (dalSPAM1 and 4)

The Naïve Bayes classifier is a simple statistical algorithm with a long history of providing surprisingly accurate results. It has been used in several spam classification studies [4, 5, 11, 9], and has become somewhat of a benchmark. It gets its name from being based on Bayes's rule of conditional probability, combined with the "naïve" assumption that all conditional probabilities are independent (given the class) [10].

Namely, to classify whether a message is spam or ham, we evaluate certain set of features of this message. If C is the message class (spam or ham) and the feature values are $f_1, \ldots, f_n$, then our goal is to calculate the probability $\mathrm{P}(C = \mathit{spam} \mid f_1, \ldots, f_n)$, while the corresponding probability $\mathrm{P}(C = \mathit{ham} \mid f_1, \ldots, f_n) = 1 - \mathrm{P}(C = \mathit{spam} \mid f_1, \ldots, f_n)$. Using Bayes' theorem, we have:

$$\mathrm{P}(C = \mathit{spam} \mid f_1, \ldots, f_n) = \frac{\mathrm{P}(C = \mathit{spam})\mathrm{P}(f_1, \ldots, f_n \mid \mathit{spam})}{\mathrm{P}(f_1, \ldots, f_n)}$$

The Naïve Bayes assumption implies further:

$$\mathrm{P}(C = \mathit{spam} \mid f_1, \ldots, f_n) = \frac{\mathrm{P}(C = \mathit{spam})\mathrm{P}(f_1 \mid \mathit{spam}) \ldots \mathrm{P}(f_n \mid \mathit{spam})}{\mathrm{P}(f_1, \ldots, f_n)}$$

Knowing further that $\mathrm{P}(f_1, \ldots, f_n) = \mathrm{P}(C = \mathit{spam})\mathrm{P}(f_1 \mid \mathit{spam}) \ldots \mathrm{P}(f_n \mid \mathit{spam}) + \mathrm{P}(C = \mathit{ham})\mathrm{P}(f_1 \mid \mathit{ham}) \ldots \mathrm{P}(f_n \mid \mathit{ham})$ makes it easy to compute $\mathrm{P}(C = \mathit{spam} \mid f_1, \ldots, f_n)$ given trained values for $\mathrm{P}(C = \mathit{spam}$ and $\mathrm{P}(C = \mathit{ham}$, and conditional probability tables for $\mathrm{P}(f_i \mid \mathit{spam})$ and $\mathrm{P}(f_i \mid \mathit{ham})$, which can be obtained from the training data using using maximum-likelihood estimation.

In terms of implementation, intermediate results are usually very small positive numbers, which can become a problem for finite precision of floating-point numbers. A solution is to convert the probabilities to their logarithms and use addition instead of multiplication. The Naïve Bayes model does not generally suffer from the sparse data problem, but zero conditional probabilities may still occur and they must be avoided. We use a smoothing technique known as a "Laplace estimator" to avoid this problem.

4 Feature Selection

One of the difficulties in any text classification task is caused by the unstructured nature of natural language which, as opposed to highly structured database items, requires a very large number of attributes to model effectively. Among the various models developed over the last few decades, the Vector Space Model (VSM) [3] is the most popular today. In VSM, each document D is represented by a vector in m dimensional term space. The underlying term T of each dimension is often referred to as a feature or attribute and represented by the weight of T in D. Earlier models use binary term weights [7], which are either 1 when the term is present in the document, or 0 when the term is absent. This weighting scheme is conceptually simple, and computationally cheap, however it lacks the ability to count on term importance—terms that are more important to a document often tend to occur more frequently in that document. This leads to the development of TF-IDF (term frequency and inverse document frequency) [6] based models, which assumes that term importance is directly proportional to the number of times a term occurs in a document, and inversely proportional to the number of documents in which it occurs. Given a document D and a term T, TF(D, T) is the number of times T occurred in D, and IDF(T) is the number of documents in which T occurred divide into the total number of documents in the dataset. Since not all documents are of the same length, TF(D, T) may turn out to be bigger in longer documents. In order to avoid this bias, we usually normalize it against the most frequent term in the document to get TFN(D, T), meanwhile, in order to avoid big number arithmetic, the base-2 logarithm of IDF(T) is usually used instead.

Since we incorporate two alternative feature representations in our filter, a "term" here can be either a naturally bounded word or a character N-gram. A character N-gram is simply any sequence of N consecutive characters taken from a text. For trigrams as an example, think of a width 3 sliding window moving from the beginning to the end of a text, one letter at a time, then the content of the window at each step is a trigram. N-gram representation has been shown competing word representation in text classification tasks [?]. For our submission, dalSPAM1 incorporates words as features and dalSPAM4 uses N-gram representation with N being 5.

One problem with both word representation and N-gram representation is the excessive number of features, which has the effect of slowing down the classification of the email as it requires more I/O to load the features and their weights into memory. In the two submissions, dalSPAM1 and dalSPAM4, one of the goals is to see if we can still achieve good accuracy while using a smaller number of features. As a result we use mutual information (MI) [8, 5, 4, 9] to determine the best 1000 features for the classifier to work with. Our parser includes the values of all header fields, as well as the entire body text (although the Document class strips out MIME attachments before parsing, leaving only the text and HTML portions of a MIME-formatted message, along with the MIME headers for each attachment). When using word representation, the characters `,.!?-:"'<>*;()[]` as well as the space, tab and new-line characters were used as token delimiters. Carriage returns were stripped out by the Document class before parsing. Tokens were truncated to a maximum length of 30 characters. For efficiency requirements of the TREC run, we do not use word stemming or stop word list. When using character N-gram representation, only alphabetical letters are considered significant, therefore all other characters are delimiters and are treated as spaces. Two or more consecutive spaces are treated as one.

5 The White-list

A white-list filter was incorporated in submissions dalSPAM1 and dalSPAM4. Each time a miss-classified ham message is corrected during training, the sender's address will be automatically added to the white-list together with a time stamp. The length limit of the white-list is controlled by a parameter, which can be customized according to different user needs. When the length limit is exceeded, oldest entries will be truncated. Users may also add entries manually if they want. An entry can be either an individual address—which takes the form “user@domain”, or a domain—which takes the form “@domain”. Manually added entries are treated as permanent (with a special time stamp), and will never get truncated.

6 Major Software Components

The Naïve Bayes submissions (Dal1 and Dal4) are based on an implementation with the following major software components:

Document	represents an RFC 822 compliant email message (RFC 822 is the basic format specification for Internet email messages);
Instance	represents an instance vector mapped from a Document;
Corpus	represents a collection of Documents;
Dictionary	stores the tokens represented by each attribute of an Instance;
Parser	parses a Document into a list of tokens;
FeatureSelector	given a set of token lists, produces a Dictionary;
Vectorizer	given a list of tokens and a Dictionary, produces an Instance;
Trainer	given a Corpus, produces a trained Classifier;
Classifier	given an Instance, returns a ”spamminess” score between 0.0 and 1.0;
Evaluator	given a set of Documents, and a set of test configurations, runs a series of cross validation experiments.

7 N-gram Based Techniques (dalSPAM2 and dalSPAM3)

Both submissions dalSPAM2 and dalSPAM3 use a simple raw byte n-gram based technique. The Dal2 and Dal3 runs used a simple raw byte n-gram-based technique. The difference is only in the n-gram size: dalSPAM2 uses n-grams of size 4, and dalSPAM3 uses n-grams of size 5. The programs maintains two files named `ham.profile` and `spam.profile` containing a list of up to 3000 byte n-grams with associated count frequencies. The Perl package was used `Text::Ngrams` to collect the n-grams. If the input message is longer than 10,000, only the first 10,000 bytes are used in order not spend too much running time on n-gram collection.

The program `classify` collects n-grams from the email file, and measure the dissimilarities of the file profile with the ham and spam profiles using the formula:

$$d = \sum_{n \in P_s \cup P_h} \left(\frac{f_1(n) - f_2(n)}{\frac{f_1(n)+f_2(n)}{2}} \right)^2 = \sum_{n \in P_s \cup P_h} \left(\frac{2 \cdot (f_1(n) - f_2(n))}{f_1(n) + f_2(n)} \right)^2 \tag{1}$$

where P_s is the set of n-grams in spam profile, and P_h is the set of n-grams in ham profile. If the dissimilarities of the e-mail file from the ham and spam profiles are s_{ham} and s_{spam}, respectively, then the score is computed using the formula:

$$score = \frac{s_{ham}}{s_{ham} + s_{spam}}.$$

The message is classified as spam if $score > 0.5$, otherwise it is classified as ham. The program classify will print the *score* and the classification result.

The program `train` updates the ham or spam profile. It reads the appropriate profile, multiply all frequencies with 0.9 (which we call *profile aging*), collect up to 3000 most frequent n-grams from the email file with count frequencies, and add this message profile to the existing profile. It will keep only up to 3000 most frequent n-grams in the profile and save it to the class profile file.

8 Evaluation

The main evaluation results on 53 submitted filters based on the TREC Spam overview report [2] are given in the following table:

Filter	Rank ROCA	Rank h=.1	Rank lam%
dalSPAM3	28	26	40
dalSPAM1	30	42	34
dalSPAM2	31	29	41
dalSPAM4	36	33	37

The reason why the lam% rank of dalSPAM3 and dalSPAM4 is significantly differs is that for those two filters we did not tune the threshold parameter, but simply used the default value of 0.5.

References

[1] *Proceedings of the Thirteenth Text REtrieval Conference (TREC-13)*, Gaithersburg, MD, USA, 2004. NIST.

[2] Gordon Cormack and Thomas Lynam. Trec 2005 spam track overview. In *Proceedings of the Fourtheenth Text REtrieval Conference (TREC-14)*, Gaithersburg, MD, USA, 2005. NIST.

[3] G. Salton et al. A vector space model for automatic indexing. *Communications of ACM*, 18(11):613–620, 1975.

[4] K. V. Chandrinos G. Paliouras I. Androutsopoulos, J. Koutsias and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, pages 9–17, Barcelona, Spain, 2000.

[5] V. Karkaletsis G. Sakkis C. D. Spyropoulos I. Androutsopoulos, G. Paliouras and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a Naïve Bayesian and a memory-based approach. In *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*, pages 1–13, Lyon, France, 2000.

[6] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*, volume 10, pages 137–142, 1999.

[7] S.O. Kimbrough and J.R. Oliver. On relevance and two aspects of the organizational memory problem. In *4th Annual Workshop on Information Technology and Systems*, pages 302–311, 1994.

[8] David Heckerman Mehran Sahami, Susan Dumais and Eric Horvitz. A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, USA, 1998. AAAI Technical Report WS-98-05.

[9] Karl-Michael Schneider. A comparison of event models for Naive Bayes anti-spam e-mail filtering. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, 2003.

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

[11] Hongjun Lu Yanlei Diao and Dekai Wu. A comparative study of classification based personal e-mail filtering. In *Proceedings of PAKDD-00, 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 408–419, Kyoto, JP, 2000. Springer Verlag.