Spam Filter: Naive Bayes and Support Vector Machine

Hongyuan Zhang & Ziwen Chen

October 2019

Abstract

Spam emails are a growing concern nowadays. In this paper, we build and compare two machine learning models for filtering out spams, a Naives Bayes model and a Support Vector Machine model (SVM). After an introduction of the spam classification problem and the two models, we demonstrate with the TREC 2007 Spam Corpus data set that both models perform well in classifying spams, with SVM slightly better than the Naive Bayes, as evaluated by measures including precision, recall, accuracy and F1 score.

1 Introduction

Email is a main way of communication in people's daily life, no matter in work or in school, or just communicating with family members. Thus, getting spam emails is becoming quite a problem because they not only occupy memory in people's computers and phones, but also waste people's time and energy to actually check them. In addition, spam emails can swindle people out of money, health and excitement. It would be nice to have a program which can automatically filter out those spam emails.

However, this is a hard problem. Because first of all, there is no "hard features" that can help the algorithm to absolutely distinguish the spam emails from normal emails. For example, there are no such words that will only appear in spam emails, and not appear in normal emails. Thus, we somehow need to construct a "soft" boundary between spam and non-spam ("ham") emails.

Second, emails are of heterogeneous structure. Some emails contain attachments like jpeg files or pdf files, while some other contain html text. In order for the algorithm to be able to "read" through those emails, we have to parse the emails using the metadata available in the emails' structure. When the email has type *multipart*, we will have to process each part separately. For *html* type, we have to get rid of the html tags and extract out the main text content, etc.

In the paper, we utilize two of the currently existing machine learning techniques to construct a automatic spam filter which can automatically parse the emails of heterogeneous structures and distinguish spam emails from ham emails. One of them is Naive Bayes, and the other one is called Support Vector Machine. Both of them are very famous machine learning techniques and have been applied in other fields successfully.

Naive Bayes constructs the spam filtering problem as a probability problem, i.e., each word is associated with a certain probability of the email being spam, and we calculate the overall probability of the email being spam using the words contained in the email. Support Vector Machine comes in with a different flavor. It constructs a hyperspace using words as dimensions, and puts each email as a point inside that hyperspace. It then tries to draw a boundary within that hyperspace to distinguish the spam points and ham points. The boundary is constructed such that the distance from the points to the line is maximized.

In this paper, we experiment with both of the techniques and run a ten-fold cross validation while tuning many parameters. All the training and validation results are displayed in the Result Analysis section.

2 Description

In this project, we implement a Naive Bayes based spam filter and a Support Vector Machine (SVM) based spam filter. Naive bayes and SVM are both supervised learning algorithms, which can be used for classification and prediction. We use the "Bag of Words" model to represent contents of each email.

2.1 Naive Bayes

The Naive Bayes method is based on Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

where A and B are events with $P(B) \neq 0$.

In the context of spam filtering, we have the following version of Bayes' Theorem:

$$P(S|W) = \frac{P(W|S)P(S)}{P(W|S)P(S) + P(W|H)P(H)} \tag{2}$$

where S is the event that a email is spam, H the event that a email is ham, W the event that a certain word appears in a given email. Thus, the equation above is literally for the probability that given that a certain word appears in a certain email, the email is a spam.

Notice that we have to guess $P(S)$, the probability that an email is spam, and $P(H)$, the probability that an email is ham. The python implementation of the Naive Bayes algorithm has an option that lets its users to input their guesses for $P(S)$ and $P(H)$ into the algorithm as parameters, or the algorithm can learn these probabilities in the process of classification.

To determine whether an email is spam or ham, the Naive Bayes method calculates $P(S|W_1, W_2, \ldots, W_n)$ for $W_1, W_2, \ldots, W_n$ in the "Bag of Words" representation of the email. To calculate $P(S|W_1, W_2, \ldots, W_n)$, we use a similar formula:

$$P(S|W_1, W_2, \ldots, W_n) = \frac{\prod_{i=1}^{n} P(W_i|S)P(S)}{\prod_{i=1}^{n} P(W_i|S)P(S) + \prod_{i=1}^{n} P(W_i|H)P(H)}$$

This equation assumes conditional independence between words, which is also why the method is named "naive". In reality, conditional independence does not often hold between words. For example, certain words may usually appear together as phrases. However, this idealized simplification in practice does not hinder the Naive Bayes method from working. In addition, an advantage of the Naive Bayes method is short computational time for training, compared to other more complex machine learning models.

2.2 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning technique that is used for classification and regression of data. Given a set of training examples, each with a classification label, belonging to one or the other of two categories, and also several features, an SVM training algorithm builds a hyperspace using the features as dimensions, and puts each training example as a point inside that hyperspace. It then tries to draw a boundary within that hyperspace to distinguish the positive and negative examples. The boundary is constructed such that the distance from the points to the line is maximized. Thus, SVM is a non-probabilistic binary linear classifier. New examples are then mapped into that same hyperspace and predicted to belong to a category based on the side of the gap on which they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

In the case of spam filtering, we use the occurrence of words in each email as their features. First, we run through all the training emails, construct a dictionary ("bad of words") out of them, and take the first k words with highest occurrence. Then we count in the email the number of appearance of each word in the dictionary. Thus email will have k features. And if we have n emails in total, our feature matrix will be $n \times k$. Then, we simply feed the feature matrix and the label vector into the SVM model and train it. After getting the trained model, we can feed it with test examples, and then the accuracy and recall of the model can be calculated.

3 Data Set

For this project, we use the TREC 2007 Spam Corpus as input data set. This data set contains 75419 emails, each with a label (ham/spam). There are 25220 hams and 50199 spams. These emails constitute all the

emails delivered to a particular server in three months of 2007. Our goal is to apply the SVM and Naive Bayes Model to differentiate hams and spams in this data set.

Each email in the data set is recorded in full details including email metadata. For the purpose of spam classification, only the information of email type is considered; other metadata such as sender/recipient are discarded in the process of constructing "Bag of Words" representation for the emails.

4 Result Analysis

We evaluate the performance of the two machine learning techniques using the traditional *precision*, *recall*, *accuracy*, and F_1 *score* measurement. *Precision* is the ratio between true positives and the sum of true positives and false positives. *Recall* is the ratio between true positives and the sum of true positives and false negatives. *Accuracy* is the ratio between the sum of true positives and true negatives and the number of items we have. F_1 is the harmonic mean of *precision* and *recall*. High precision intuitively means that, in our case, among all emails that our model predicts to be spam, a high percentage of those emails are true spams. High recall intuitively means that, in our case, a high percentage of all true spam emails are predicted by our model correctly to be spams. Accuracy concerns more about the fraction of correct predictions. We want all four measurements to be as high as possible. In the tables and plot below, we display the experimentation results for both of the machine learning techniques, with each experiment using ten-fold cross validation. We examine the effectiveness of the two models across various email collection sizes. We first use only 500 emails in total for training and testing the Naive Bayes Model, and then use the same set of 500 emails to train and test the SVM Model. We repeat this process for email collection sizes of 1,000, 10,000 and 70,000.

size	500	1000	10000	70000
precision	0.8777	0.8903	0.9920	0.9861
recall	1.0	1.0	0.9624	0.9708
accuracy	0.8860	0.9040	0.9657	0.9718
F1 score	0.9339	0.9409	0.9769	0.9784

Table 1: Experiment result for Naive Bayes

size	500	1000	10000	70000
precision	0.9351	0.9646	0.9905	0.9899
recall	0.9972	0.9988	0.9984	0.9960
accuracy	0.942	0.97	0.9916	0.991
F1 score	0.9646	0.9811	0.9944	0.9929

Table 2: Experiment result for SVM

As we can readily tell from Table 1, Table 2, and Figure 1, both models perform generally well in classifying spams. However, for all data sizes, SVM has higher accuracy and F1 scores than the Naive Bayes. The Naive Bayes, particularly when the data size is small, is not as precise as SVM. Nontheless, given large a data size, Naive Bayes can have the same order of magnitude of precision as SVM. In addition, both models have fairly high recall, though the Naive Bayes has more fluctuated recall across different data sizes whereas SVM maintains high recall for all data sizes.

5 Discussion

From our analysis of results above, model evaluation measures indicate that SVM is a better model for classifying spams in our data set than the Naive Bayes is. However, our conclusion here may not be applied to the general case, i.e. the SVM model may not always be superior to the Naive Bayes model when it comes to the problem of spam classification. Moreover, considering the potential datedness of our data set, our conclusion may not be generalized to spam emails nowadays. It is possible that recent spam emails are "smarter" in hiding their spam smells and traces.

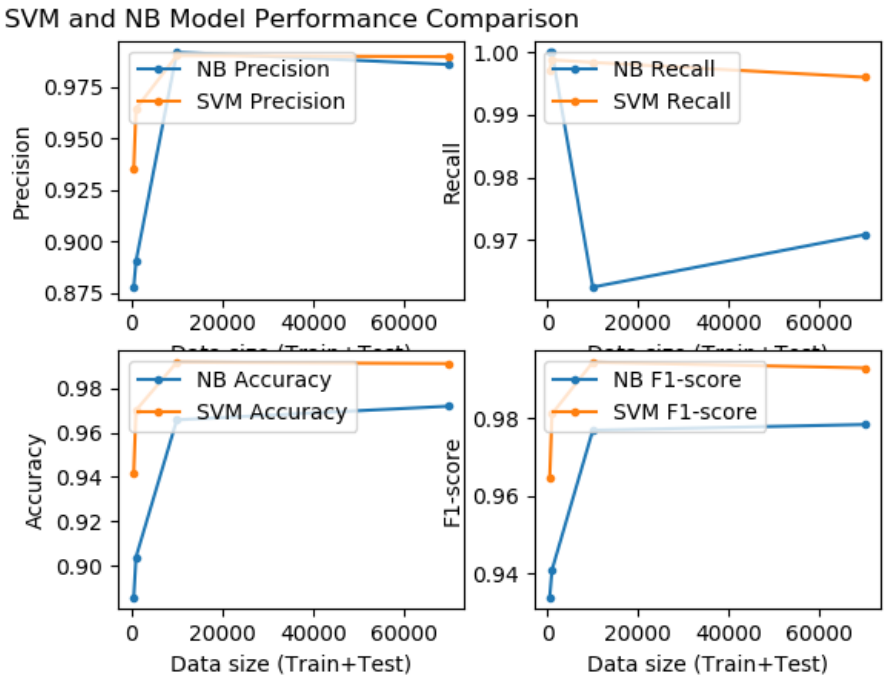

Figure 1: NB v.s. SVM Performance Comparison: Precision, Recall, Accuracy, F1 score

Future work should focus on applying these two models to larger data sets of emails to generalize the conclusion. The data sets should be preferably new and can represent spam emails in these days. Further, in our work, we did not consider analysis of multimedia contents, which can be important features for the purpose of spam identification. Examination of multimedia contents and attachments may help in our context, as well as investigation of metadata of emails. Future work can focus on incorporating these components of an email into models, which can hopefully improve our models.